III YEAR – VI SEMESTER
COURSE CODE: 4BIT6C3
CORE COURSE XIV – WEB PROGRAMMING

HTML Introduction

What is HTML?

HTML is a language for describing web pages.

❖ HTML stands for **H**yper **T**ext **M**arkup **L**anguage

❖ HTML is not a programming language, it is a markup language

❖ A markup language is a set of markup tags

❖ HTML uses markup tags to describe web pages

HTML Tags

HTML markup tags are usually called HTML tags

❖HTML tags are keywords surrounded by angle brackets like <html>

❖HTML tags normally come in pairs like <b> and **</b>**

❖The first tag in a pair is the start tag, the second tag is the end tag

❖Start and end tags are also called opening tags and closing tags

Department of Information Technology

APSA College Tiruppattur-630211

HTML Documents = Web Pages

HTML documents **describe web pages**

HTML documents contain **HTML tags** and **plain text**

HTML documents are also **called web pages**

The purpose of a web browser (like Internet Explorer or Firefox) is to read HTML documents and display them as web pages. *The browser does not display the HTML tags, but uses the tags to* interpret the content of the page:

```
<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>
```

Example Explained

The text between <html> and </html> describes the web page

The text between <body> and </body> is the visible page content

The text between <h1> and </h1> is displayed as a heading

The text between <p> and </p> is displayed as a paragraph

HTML Headings

HTML headings are defined with the **<h1>** to **<h6>** tags

Example

<h1>This is a heading</h1>

<h2>This is a heading</h2>

<h3>This is a heading</h3> .

HTML Paragraphs

HTML paragraphs are defined with the <p> tag.

Example

<p>This is a paragraph.</p>

<p>This is another paragraph.</p>

HTML Links

HTML links are defined with the <a> tag.

Example

```
<a href="http://www.apsacollege.com">This is a link</a>
```

HTML Images

HTML images are defined with the <img> tag.

Example

```
<img src=“abc.jpg" width="104" height="142" />
```

HTML Comments

Comments can be inserted into the HTML code to make it more readable and understandable. Comments are ignored by the browser and are not displayed.

Example

```
<!-- This is a comment -->
```

HTML Formatting Tags

Tag	Description
<b>	Defines bold text
<big>	Defines big text
<em>	Defines emphasized text
<i>	Defines italic text
<small>	Defines small text
<strong>	Defines strong text
<sub>	Defines subscripted text
<sup>	Defines superscripted text
<ins>	Defines inserted text
<del>	Defines deleted text

```
<html>
<body>
<p><b>This text is bold</b></p>
<p><strong>This text is strong</strong></p>
<p><big>This text is big</big></p>
<p><em>This text is emphasized</em></p>
<p><i>This text is italic</i></p>
<p><small>This text is small</small></p>
<p>This is<sub> subscript</sub> and
<sup>superscript</sup></p>
</body>
</html>
```

A text with a deleted part and a new inserted part:

```
<p>My favorite color is <del>blue</del> <ins>red</ins>!</p>
```

HTML Attributes

HTML elements can have attributes

❖Attributes provide additional information about an element

❖Attributes are always specified in the start tag

❖Attributes come in name/value pairs like: name="value"

HTML Lines

The <hr /> tag creates a horizontal line in an HTML page.

The hr element can be used to separate content:

Example

```
<p>This is a paragraph</p>

<hr />

<p>This is a paragraph</p>

<hr />

<p>This is a paragraph</p>
```

hr Attribute Values Syntax

<hr align="value" />

Value	Description
left	Left-aligns the horizontal line
center	Center-aligns the horizontal line (this is default)
right	Right-aligns the horizontal line

Example

A left-aligned horizontal line:

<hr align="left" width="50%" />

HTML Line Breaks

Use the <br /> tag if you want a line break (a new line) without starting a new paragraph:

Example

<p>This is**<br />**a para**<br />**graph with line breaks</p>

HTML <abbr> Tag

The <abbr> tag describes an abbreviated phrase.

By marking up abbreviations you can give useful information to browsers, spellcheckers, screen readers, translation systems and search-engines.

Example

An abbreviation is marked up as follows:

The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.

HTML <acronym> Tag

An acronym can be spoken as if it were a word, example NATO, NASA, ASAP, GUI.

By marking up acronyms you can give useful information to browsers, spellcheckers, screen readers, translation systems and search-engines.

Example

An acronym is marked up as follows:

Can I get this <acronym title="as soon as possible">ASAP</acronym>?

HTML <address> Tag

The <address> tag defines the contact information for the author or owner of a document.

This way, the reader is able to contact the document's owner.

The address element is usually added to the header or footer of a webpage.

Example

```
Contact information for APSAC:
<address>
Written by Ananthan<br />
<a href="mailto:ananthakrishnan@apsacollege.com">Email us</a><br />
Address: No : 564, TPR <br />
Phone: 12 34 56 78
</address>
```

HTML <bdo> Tag

bdo stand for bidirectional override.

The <bdo> tag allows you to specify the text direction and override the bidirectional algorithm.

Example

Specify the text direction:

<bdo dir="rtl">Here is some Hebrew text!</bdo>

HTML <blockquote> Tag

The **<blockquote>** tag defines a long quotation.

A browser inserts white space before and after a blockquote element. It also insert margins for the blockquote element.

Example

<blockquote>

Here is a long quotation here is a long quotation here is a long quotation here is a long quotation here is a long quotation here is a long quotation here is a long quotation here is a long quotation here is a long quotation.

</blockquote>

HTML <q> Tag

The **<q>** tag defines a short quotation.

The browser will insert quotation marks around the quotation.

Example

A short quotation is marked up as follows:

<q>

Here is a short quotation here is a short quotation

</q>

HTML Hyperlinks (Links)

A hyperlink (or link) is a word, group of words, or image that you can click on to jump to a new document or a new section within the current document.When you move the cursor over a link in a Web page, the arrow will turn into a little hand.

Links are specified in HTML using the <a> tag.

The <a> tag can be used in two ways:

❖**To create a link to another document, by using the href attribute**

❖**To create a bookmark inside a document, by using the name attribute**

HTML Link Syntax

The HTML code for a link is simple. It looks like this:

<a href="url">Link text</a>

The href attribute specifies the **destination of a link**.

The "Link text" doesn't have to be text. You can link from an **image** or **any other HTML element.**

Example

<a href="http://www.apsacollege.com/">Visit APSAC</a>

HTML Links - The target Attribute

The **target attribute** specifies where to open the linked document.

The example below will open the linked document in a new browser window:

Example

<a href="http://www.apsacollege.com/" target="_blank">Visit APSAC</a>

HTML Links - The name Attribute

The name attribute specifies the name of an anchor.

The name attribute is used to create a **bookmark inside an HTML document**.

Bookmarks are not displayed in any special way. They are invisible to the reader.

Example

A named anchor inside an HTML document:

<a name="tips">Useful Tips Section</a>

Create a link to the "Useful Tips Section" inside the same document:

<a href="#tips">Visit the Useful Tips Section</a>

The HTML Style Attribute

The purpose of the style attribute is:

To provide a common way to style all HTML elements.

Styles was introduced with HTML 4, as the new and preferred way to style HTML elements. With HTML styles, styles can be added to HTML elements directly by using the style attribute, or indirectly in separate style sheets (CSS files).

Tags	Description
<center>	Defines centered content
<font>	Defines HTML fonts
<s> and <strike>	Defines strikethrough text
<u>	Defines underlined text

Attributes	Description
align	Defines the alignment of text
bgcolor	Defines the background color
color	Defines the text color

HTML Style Example - Background Color

The background-color property defines the background color for an element:

Example

```
<html>

<body style="background-color:yellow">

<h2 style="background-color:red">This is a heading</h2>

<p style="background-color:green">This is a paragraph.</p>

</body>

</html>
```

HTML Style Example - Font, Color and Size

The font-family, color, and font-size properties defines the font, color, and size of the text in an element:

Example

```
<html>
<body>
<h1 style="font-family:verdana">A heading</h1>
<p style="font-family:arial;color:red;font-size:20px;">
A paragraph.</p>
</body>
</html>
```

HTML Style Example - Text Alignment

The text-align property specifies the horizontal alignment of text in an element:

Example

```
<html>

<body>

<h1 style="text-align:center">This is a heading</h1>

<p>The heading above is aligned to the center of this page.</p>

</body>

</html>
```

HTML The <img> Tag and the Src Attribute

In HTML, images are defined with the <img> tag.

The <img> tag is empty, which means that it contains attributes only, and has no closing tag.

To display an image on a page, you need to use the **src** attribute. Src stands for "source". The value of the src attribute is the URL of the image you want to display.

Syntax for defining an image:

<img src="url" alt="some_text"/>

HTML The Alt Attribute

The required alt attribute specifies an alternate text for an image, if the image cannot be displayed.

The value of the alt attribute is an author-defined text:

<img src="boat.gif" alt="Big Boat" />

The alt attribute provides alternative information for an image if a user for some reason cannot view it (because of slow connection, an error in the src attribute, or if the user uses a screen reader).

HTML Tables

Tables are defined with the <table> tag.

A table is divided into **rows (with the <tr> tag),** and each row is divided into data cells (with the <td> tag). **td stands for "table data,"** and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, other tables, etc.

Apples	**44%**
Bananas	**23%**
Oranges	**13%**
Other	**10%**

Table Example

```
<table>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

$\longrightarrow$

row 1, cell 1	**row 1, cell 2**
row 2, cell 1	**row 2, cell 2**

HTML Tables and the Border Attribute

If you do not specify a border attribute, the table will be displayed without borders. Sometimes this can be useful, but most of the time, we want the borders to show.

To display a table with borders, specify the border attribute:

```
<table border=“1”>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```

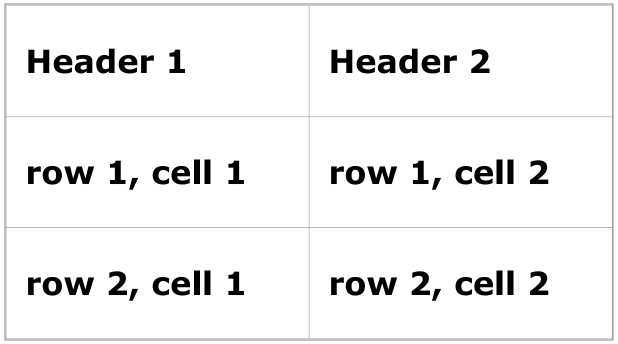
row 1, cell 1	**row 1, cell 2**
row 2, cell 1	**row 2, cell 2**

HTML Table Headers

Header information in a table are defined with the <th> tag.

The text in a **th** element will be bold and centered.

```
<table border="1">
<tr>
<th>Header 1</th>
<th>Header 2</th>
</tr>
<tr>
<td>row 1, cell 1</td>
<td>row 1, cell 2</td>
</tr>
<tr>
<td>row 2, cell 1</td>
<td>row 2, cell 2</td>
</tr>
</table>
```



Header 1	Header 2
row 1, cell 1	row 1, cell 2
row 2, cell 1	row 2, cell 2

Column span

```
<html>
<body>
<h4>Cell that spans two columns:</h4>
<table border="1">
<tr>
  <th>Name</th>
  <th colspan="2">Telephone</th>
</tr>
<tr>
  <td>Ananthan</td>
  <td>555 77 854</td>
  <td>555 77 855</td>
</tr>
</table>
</body>
</html>
```

Cell that spans two columns:

Name	Telephone	
Ananthan	555 77 854	555 77 855

rowspan

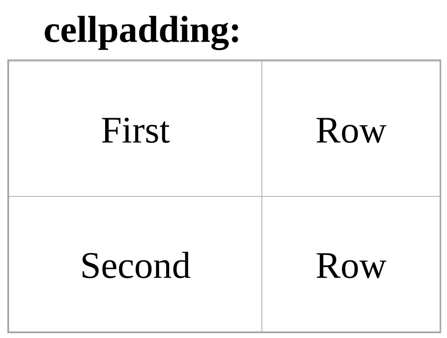
```
<body>
<html>
<h4>Cell that spans two rows:</h4>
<table border="1">
<tr>
  <th>First Name:</th>
  <td> Ananthan </td>
</tr>
<tr>
  <th rowspan="2">Telephone:</th>
  <td>555 77 854</td>
</tr>
<tr>
  <td>555 77 855</td>
</tr>
</table>
</body>
</html>
```

Cell that spans two rows:

First Name:	Ananthan
Telephone:	555 77 854
	555 77 855

cellpadding

```
<html>
<body>
<h4>With cellpadding:</h4>
<table border="1"  cellpadding="20">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
</body>
```



cellpadding:

First	Row
Second	Row

cellspacing

```
<html>
<body>
<h4>With cellspacing:</h4>
<table border="1“ cellspacing="20">
<tr>
  <td>First</td>
  <td>Row</td>
</tr>
<tr>
  <td>Second</td>
  <td>Row</td>
</tr>
</table>
</body>
</html>
```

```
<html>
<head>
<style type="text/css">
thead {color:green}
tbody {color:blue;height:50px}
tfoot {color:red}
</style>
</head>
<body>
<table border="1">
 <thead>
  <tr>
   <th>Month</th>
   <th>Savings</th>
  </tr>
 </thead>
<tfoot>
  <tr>
   <td>Sum</td>
   <td>$180</td>
  </tr>
 </tfoot>
 <tbody>
  <tr>
   <td>January</td>
   <td>$100</td>
  </tr>
  <tr>
   <td>February</td>
   <td>$80</td>
  </tr>
 </tbody>
</table>
</body>
</html>
```

HTML Lists

An ordered list:

1. **The first list item**
2. **The second list item**
3. **The third list item**

An unordered list:

- **List item**
- **List item**
- **List item**

HTML Unordered Lists

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items are marked with bullets (typically small black circles).

```
<ul>
<li>Coffee</li>
<li>Milk</li>
</ul>
```

looks in a browser:

- Coffee
- Milk

HTML Ordered Lists

An ordered list starts with the **<ol>** tag. Each list item starts with the **<li>** tag.

The list items are marked with numbers.

```
<ol>
<li>Coffee</li>
<li>Milk</li>
</ol>
```

looks in a browser:

1.Coffee

2.Milk

Different types of ordered lists

```
<html>

<body>

<h4>Numbered list:</h4>

<ol>
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>


<h4>Letters list:</h4>

<ol type="A">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>

<h4>Lowercase letters list:</h4>

<ol type="a">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>

<h4>Roman numbers list:</h4>

<ol type="I">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>

<h4>Lowercase Roman numbers list:</h4>

<ol type="i">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>


</body>

</html>
```

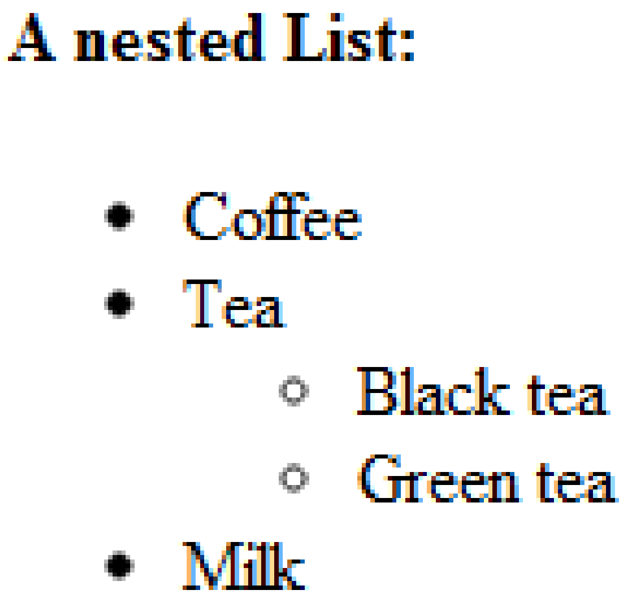
Different types of unordered lists

```
<html>
<body>
<h4>Disc bullets list:</h4>
<ul type="disc">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
```

```
<h4>Circle bullets list:</h4>
<ul type="circle">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>
```

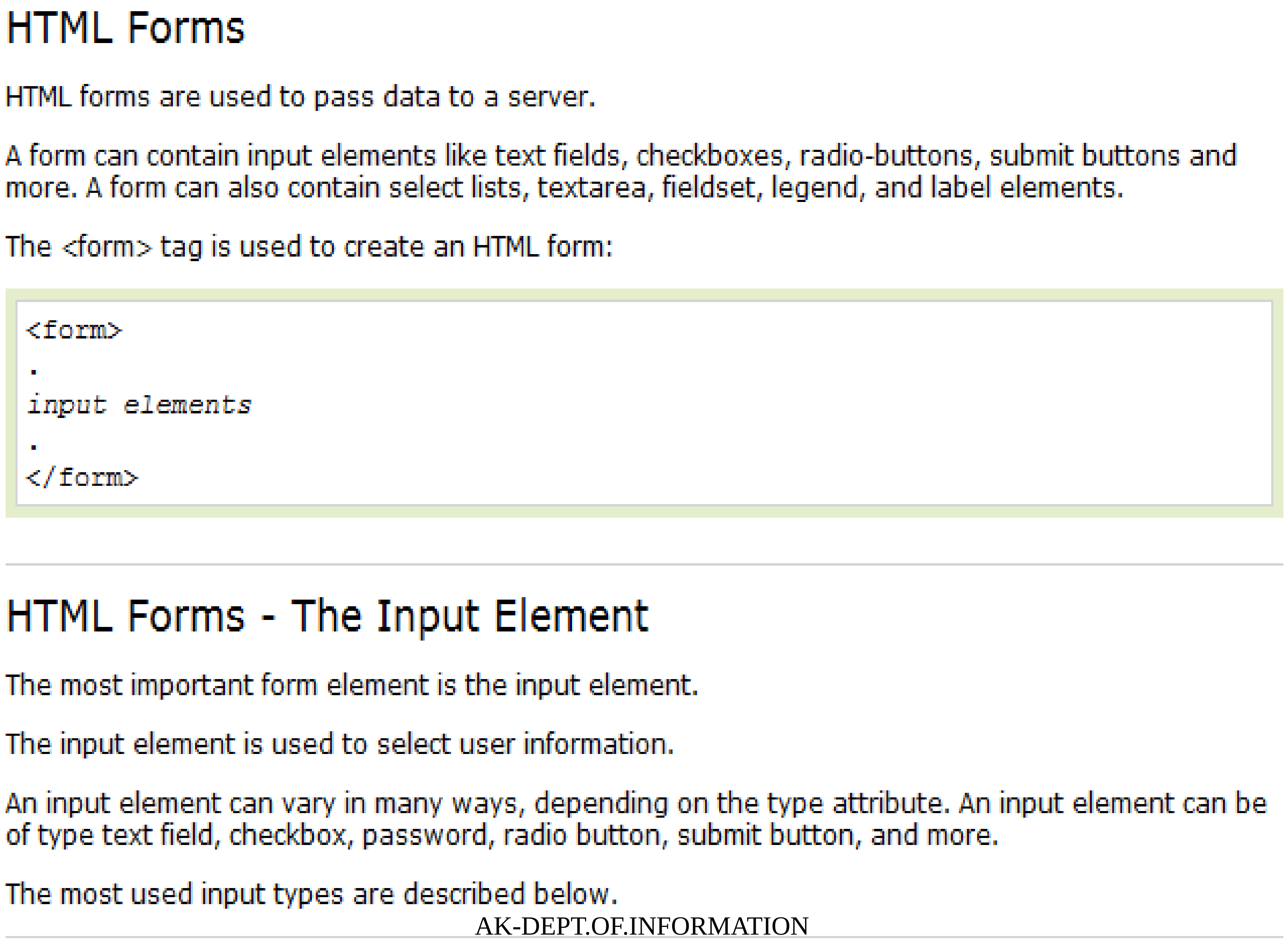
```
<h4>Square bullets list:</h4>
<ul type="square">
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>

</body>
</html>
```

```
<html>
<body>
<h4>A nested List:</h4>
<ul>
  <li>Coffee</li>
  <li>Tea
    <ul>
    <li>Black tea</li>
    <li>Green tea</li>
    </ul>
  </li>
  <li>Milk</li>
</ul>
</body>
</html>
```



A nested List:

- Coffee
- Tea
  - Black tea
  - Green tea
- Milk



HTML Forms

HTML forms are used to pass data to a server.

A form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and more. A form can also contain select lists, textarea, fieldset, legend, and label elements.

The <form> tag is used to create an HTML form:

```
<form>
.
input elements
.
</form>
```

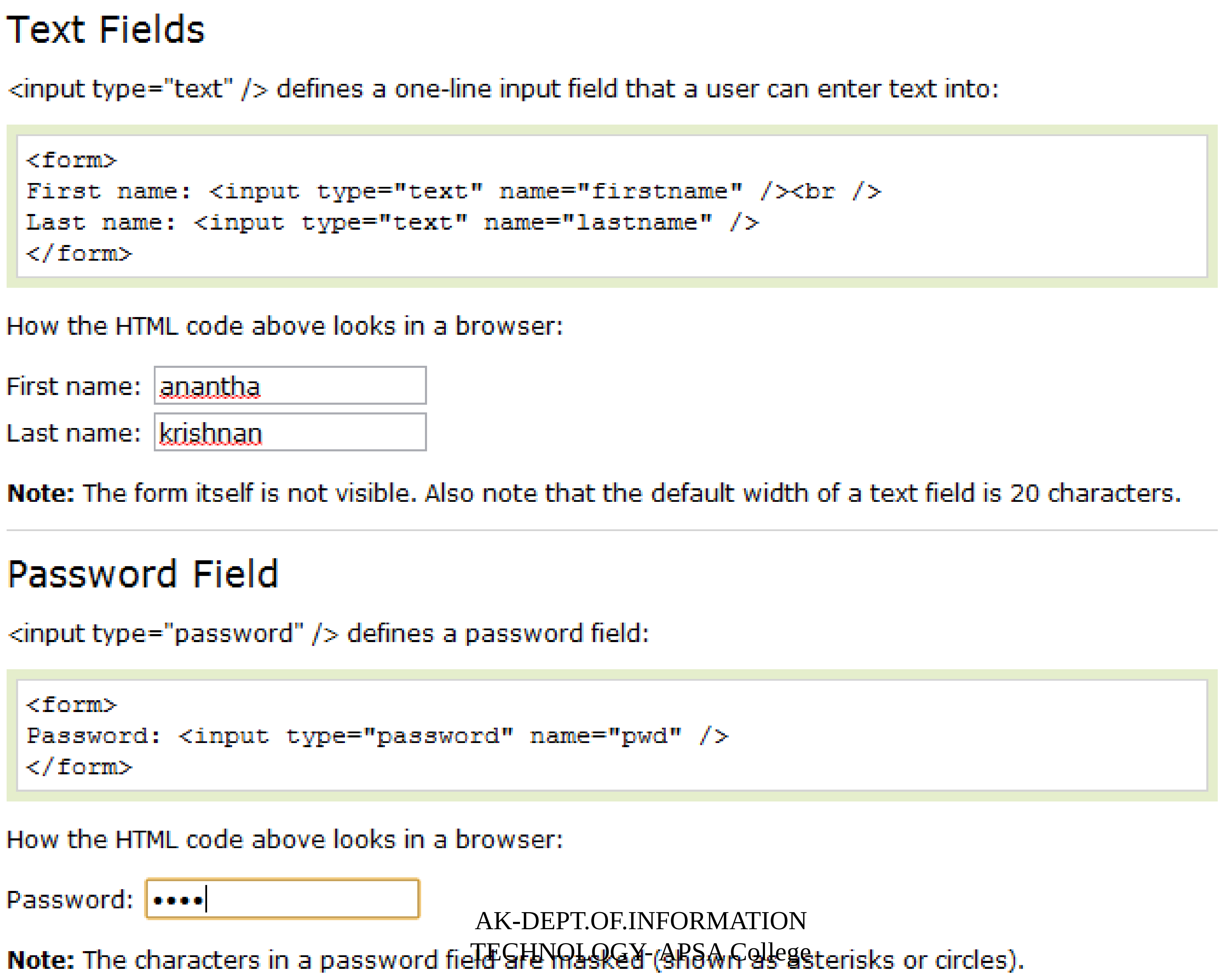
HTML Forms - The Input Element

The most important form element is the input element.

The input element is used to select user information.

An input element can vary in many ways, depending on the type attribute. An input element can be of type text field, checkbox, password, radio button, submit button, and more.

The most used input types are described below.



Text Fields

<input type="text" /> defines a one-line input field that a user can enter text into:

```
<form>
First name: <input type="text" name="firstname" /><br />
Last name: <input type="text" name="lastname" />
</form>
```

How the HTML code above looks in a browser:

First name: anantha

Last name: krishnan

Note: The form itself is not visible. Also note that the default width of a text field is 20 characters.

Password Field

<input type="password" /> defines a password field:

```
<form>
Password: <input type="password" name="pwd" />
</form>
```

How the HTML code above looks in a browser:

Password: ••••

Note: The characters in a password field are masked (shown as asterisks or circles).

Radio Buttons

<input type="radio" /> defines a radio button. Radio buttons let a user select ONLY ONE one of a limited number of choices:

```
<form>
<input type="radio" name="sex" value="male" /> Male<br />
<input type="radio" name="sex" value="female" /> Female
</form>
```

How the HTML code above looks in a browser:

○ Male
○ Female

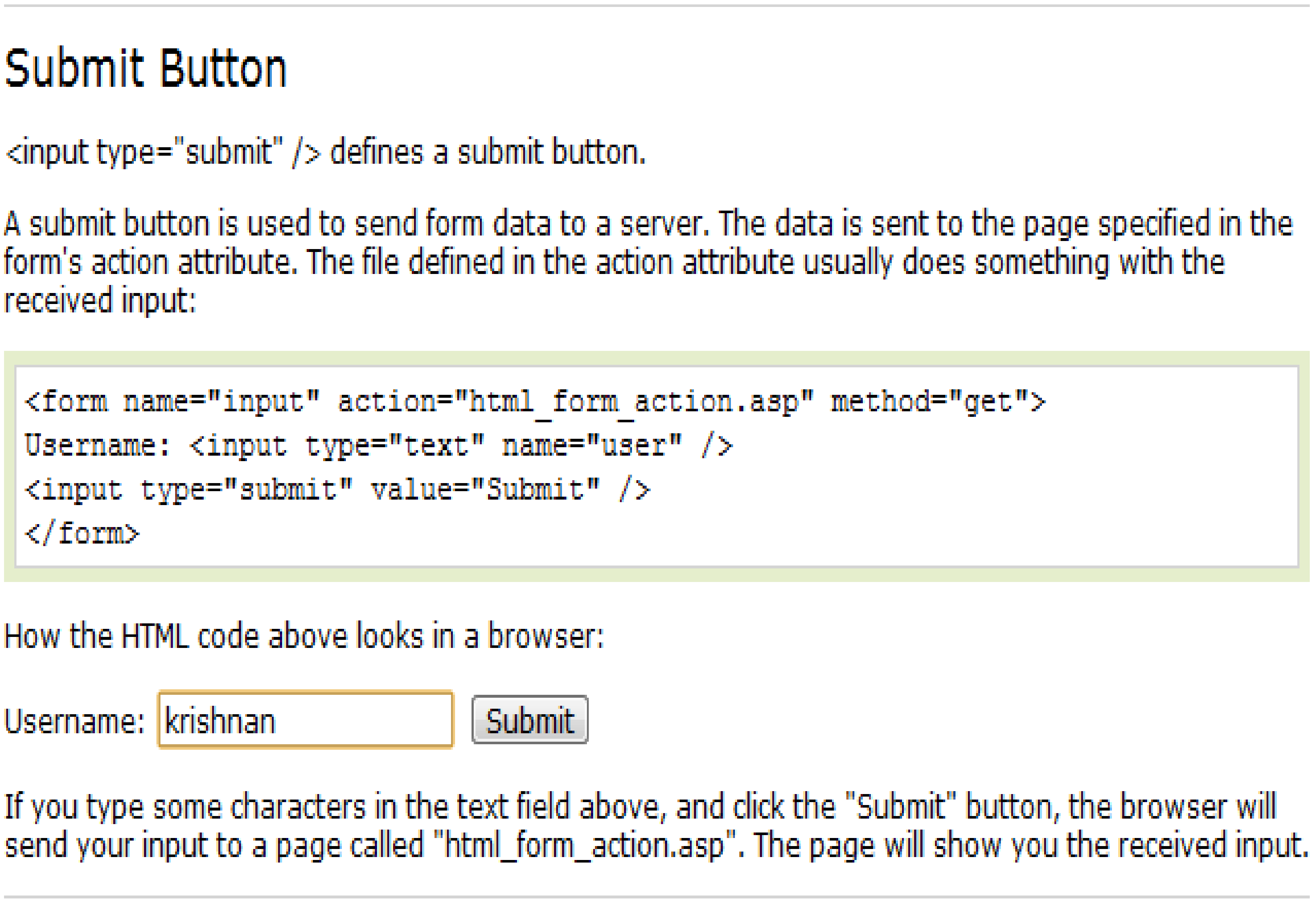
Checkboxes

<input type="checkbox" /> defines a checkbox. Checkboxes let a user select ONE or MORE options of a limited number of choices.

```
<form>
<input type="checkbox" name="vehicle" value="Bike" /> I have a bike<br />
<input type="checkbox" name="vehicle" value="Car" /> I have a car
</form>
```

How the HTML code above looks in a browser:

☐ I have a bike
☐ I have a car



Submit Button

<input type="submit" /> defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually does something with the received input:

```
<form name="input" action="html_form_action.asp" method="get">
Username: <input type="text" name="user" />
<input type="submit" value="Submit" />
</form>
```

How the HTML code above looks in a browser:

Username: krishnan

If you type some characters in the text field above, and click the "Submit" button, the browser will send your input to a page called "html_form_action.asp". The page will show you the received input.

HTML Frames

With frames, you can display more than one HTML document in the same browser window. Each HTML document is called a frame, and each frame is independent of the others.

The disadvantages of using frames are:

- The web developer must keep track of more HTML documents
- It is difficult to print the entire page

The HTML frameset Element

The frameset element holds one or more frame elements. Each frame element can hold a separate document.

The frameset element states HOW MANY columns or rows there will be in the frameset, and HOW MUCH percentage/pixels of space will occupy each of them.

The HTML frame Element

The <frame> tag defines one particular window (frame) within a frameset.

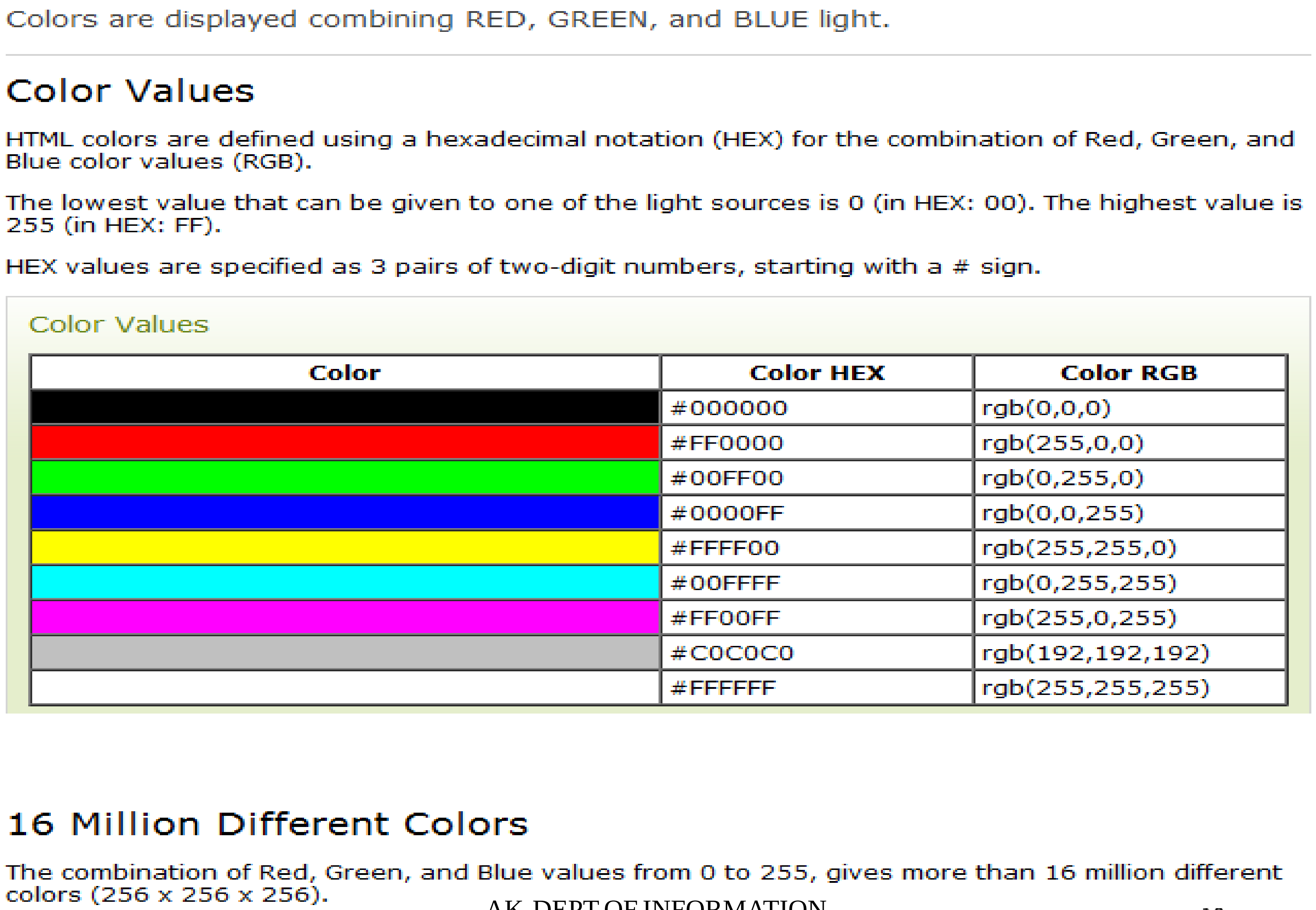
In the example below we have a frameset with two columns.

The first column is set to 25% of the width of the browser window. The second column is set to 75% of the width of the browser window. The document "frame_a.htm" is put into the first column, and the document "frame_b.htm" is put into the second column:

```
<frameset cols="25%,75%">
   <frame src="frame_a.htm" />
   <frame src="frame_b.htm" />
</frameset>
```

Note: The frameset column size can also be set in pixels (cols="200,500"), and one of the columns can be set to use the remaining space, with an asterisk (cols="25%,*").

HTML Colors



Colors are displayed combining RED, GREEN, and BLUE light.

Color Values

HTML colors are defined using a hexadecimal notation (HEX) for the combination of Red, Green, and Blue color values (RGB).

The lowest value that can be given to one of the light sources is 0 (in HEX: 00). The highest value is 255 (in HEX: FF).

HEX values are specified as 3 pairs of two-digit numbers, starting with a # sign.

Color Values

Color	Color HEX	Color RGB
⬛	#000000	rgb(0,0,0)
🟥	#FF0000	rgb(255,0,0)
🟩	#00FF00	rgb(0,255,0)
🟦	#0000FF	rgb(0,0,255)
🟨	#FFFF00	rgb(255,255,0)
🟦	#00FFFF	rgb(0,255,255)
🟪	#FF00FF	rgb(255,0,255)
▨	#C0C0C0	rgb(192,192,192)
⬜	#FFFFFF	rgb(255,255,255)

16 Million Different Colors

The combination of Red, Green, and Blue values from 0 to 255, gives more than 16 million different colors (256 x 256 x 256).

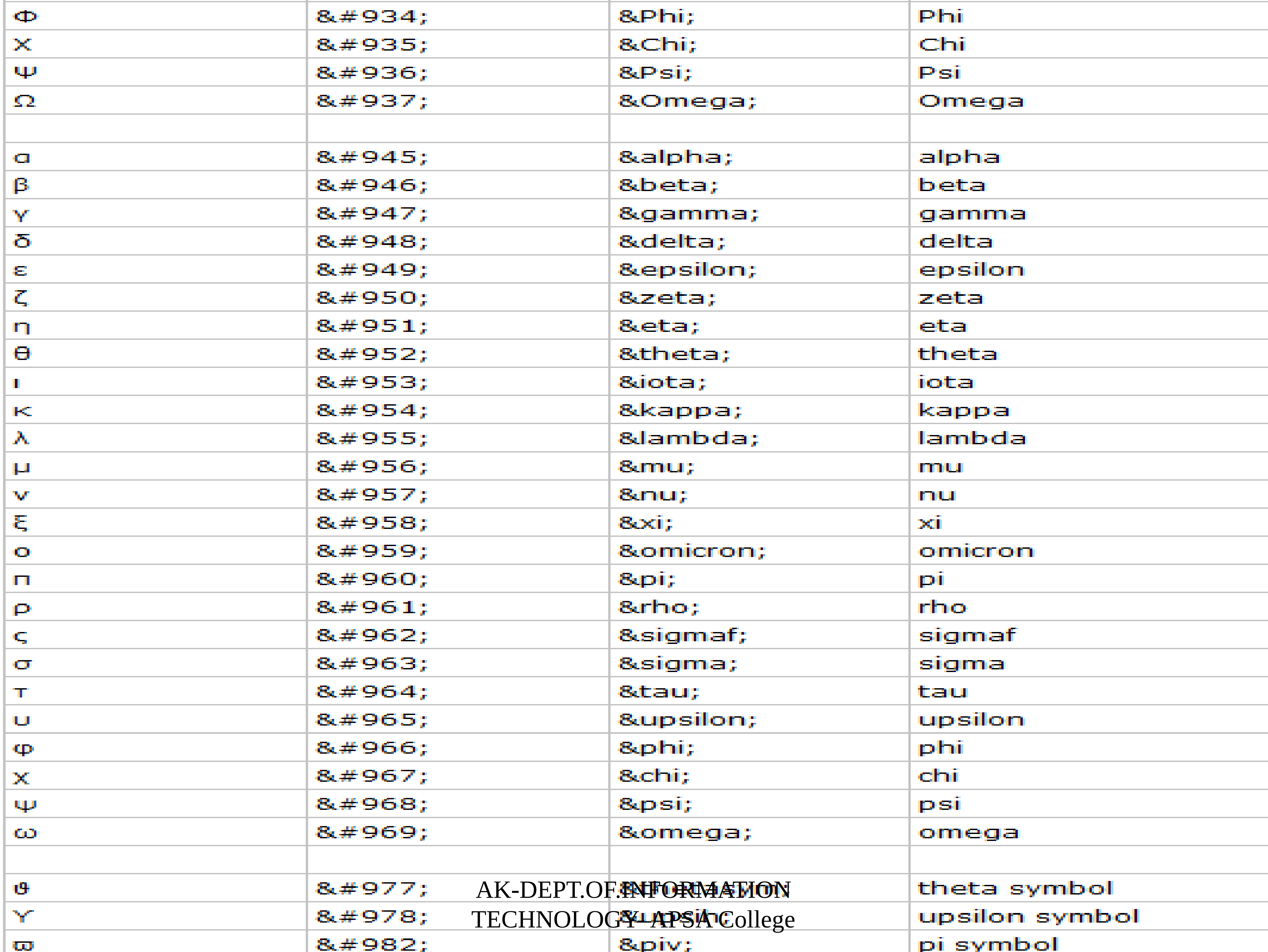
Math Symbols Supported by HTML

Character	Entity Number	Entity Name	Description
∀	&#8704;	&forall;	for all
∂	&#8706;	&part;	part
∃	&#8707;	&exist;	exists
∅	&#8709;	&empty;	empty
∇	&#8711;	&nabla;	nabla
∈	&#8712;	&isin;	isin
∉	&#8713;	&notin;	notin
∋	&#8715;	&ni;	ni
∏	&#8719;	&prod;	prod
∑	&#8721;	&sum;	sum
−	&#8722;	&minus;	minus
∗	&#8727;	&lowast;	lowast
√	&#8730;	&radic;	square root
∝	&#8733;	&prop;	proportional to
∞	&#8734;	&infin;	infinity
∠	&#8736;	&ang;	angle
∧	&#8743;	&and;	and
∨	&#8744;	&or;	or
∩	&#8745;	&cap;	cap
∪	&#8746;	&cup;	cup
∫	&#8747;	&int;	integral
∴	&#8756;	&there4;	therefore
∼	&#8764;	&sim;	similar to
≅	&#8773;	&cong;	congruent to

≈	&#8776;	&asymp;	almost equal
≠	&#8800;	&ne;	not equal
≡	&#8801;	&equiv;	equivalent
≤	&#8804;	&le;	less or equal
≥	&#8805;	&ge;	greater or equal
⊂	&#8834;	&sub;	subset of
⊃	&#8835;	&sup;	superset of
⊄	&#8836;	&nsub;	not subset of
⊆	&#8838;	&sube;	subset or equal
⊇	&#8839;	&supe;	superset or equal
⊕	&#8853;	&oplus;	circled plus
⊗	&#8855;	&otimes;	cirled times
⊥	&#8869;	&perp;	perpendicular
⋅	&#8901;	&sdot;	dot operator

Greek Letters Supported by HTML

Character	Entity Number	Entity Name	Description
Α	&#913;	&Alpha;	Alpha
Β	&#914;	&Beta;	Beta
Γ	&#915;	&Gamma;	Gamma
Δ	&#916;	&Delta;	Delta
Ε	&#917;	&Epsilon;	Epsilon
Ζ	&#918;	&Zeta;	Zeta
Η	&#919;	&Eta;	Eta
Θ	&#920;	&Theta;	Theta
Ι	&#921;	&Iota;	Iota
Κ	&#922;	&Kappa;	Kappa
Λ	&#923;	&Lambda;	Lambda
Μ	&#924;	&Mu;	Mu
Ν	&#925;	&Nu;	Nu
Ξ	&#926;	&Xi;	Xi
Ο	&#927;	&Omicron;	Omicron
Π	&#928;	&Pi;	Pi
Ρ	&#929;	&Rho;	Rho
	undefined		Sigmaf
Σ	&#931;	&Sigma;	Sigma
Τ	&#932;	&Tau;	Tau
Υ	&#933;	&Upsilon;	Upsilon



Φ	&#934;	&Phi;	Phi
Χ	&#935;	&Chi;	Chi
Ψ	&#936;	&Psi;	Psi
Ω	&#937;	&Omega;	Omega
α	&#945;	&alpha;	alpha
β	&#946;	&beta;	beta
γ	&#947;	&gamma;	gamma
δ	&#948;	&delta;	delta
ε	&#949;	&epsilon;	epsilon
ζ	&#950;	&zeta;	zeta
η	&#951;	&eta;	eta
θ	&#952;	&theta;	theta
ι	&#953;	&iota;	iota
κ	&#954;	&kappa;	kappa
λ	&#955;	&lambda;	lambda
μ	&#956;	&mu;	mu
ν	&#957;	&nu;	nu
ξ	&#958;	&xi;	xi
ο	&#959;	&omicron;	omicron
π	&#960;	&pi;	pi
ρ	&#961;	&rho;	rho
ς	&#962;	&sigmaf;	sigmaf
σ	&#963;	&sigma;	sigma
τ	&#964;	&tau;	tau
υ	&#965;	&upsilon;	upsilon
φ	&#966;	&phi;	phi
χ	&#967;	&chi;	chi
ψ	&#968;	&psi;	psi
ω	&#969;	&omega;	omega
ϑ	&#977;	&thetasym;	theta symbol
ϒ	&#978;	&upsih;	upsilon symbol
ϖ	&#982;	&piv;	pi symbol

Other Entities Supported by HTML

Character	Entity Number	Entity Name	Description
Œ	&#338;	&OElig;	capital ligature OE
œ	&#339;	&oelig;	small ligature oe
Š	&#352;	&Scaron;	capital S with caron
š	&#353;	&scaron;	small S with caron
Ÿ	&#376;	&Yuml;	capital Y with diaeres
ƒ	&#402;	&fnof;	f with hook
ˆ	&#710;	&circ;	modifier letter circumflex accent
˜	&#732;	&tilde;	small tilde
	&#8194;	&ensp;	en space
	&#8195;	&emsp;	em space
	&#8201;	&thinsp;	thin space
	&#8204;	&zwnj;	zero width non-joiner
	&#8205;	&zwj;	zero width joiner
	&#8206;	&lrm;	left-to-right mark
	&#8207;	&rlm;	right-to-left mark
–	&#8211;	&ndash;	en dash
—	&#8212;	&mdash;	em dash
‘	&#8216;	&lsquo;	left single quotation mark
’	&#8217;	&rsquo;	right single quotation mark
‚	&#8218;	&sbquo;	single low-9 quotation mark
“	&#8220;	&ldquo;	left double quotation mark
”	&#8221;	&rdquo;	right double quotation mark
„	&#8222;	&bdquo;	double low-9 quotation mark
†	&#8224;	&dagger;	dagger
‡	&#8225;	&Dagger;	double dagger
•	&#8226;	&bull;	bullet
…	&#8230;	&hellip;	horizontal ellipsis
‰	&#8240;	&permil;	per mille
′	&#8242;	&prime;	minutes
″	&#8243;	&Prime;	seconds
‹	&#8249;	&lsaquo;	single left angle quotation
›	&#8250;	&rsaquo;	single right angle quotation
‾	&#8254;	&oline;	overline
€	&#8364;	&euro;	euro

	&#8243;	&Prime;	seconds
‹	&#8249;	&lsaquo;	single left angle quotation
›	&#8250;	&rsaquo;	single right angle quotation
‾	&#8254;	&oline;	overline
€	&#8364;	&euro;	euro
™	&#8482;	&trade;	trademark
←	&#8592;	&larr;	left arrow
↑	&#8593;	&uarr;	up arrow
→	&#8594;	&rarr;	right arrow
↓	&#8595;	&darr;	down arrow
↔	&#8596;	&harr;	left right arrow
↵	&#8629;	&crarr;	carriage return arrow
⌈	&#8968;	&lceil;	left ceiling
⌉	&#8969;	&rceil;	right ceiling
⌊	&#8970;	&lfloor;	left floor
⌋	&#8971;	&rfloor;	right floor
◊	&#9674;	&loz;	lozenge
♠	&#9824;	&spades;	spade
♣	&#9827;	&clubs;	club
♥	&#9829;	&hearts;	heart
♦	&#9830;	&diams;	diamond

The HTML meta Element

Metadata is information about data.

The <meta> tag provides metadata about the HTML document. Metadata will not be displayed on the page, but will be machine parsable.

Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.

The <meta> tag always goes inside the head element.

The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.

Keywords for Search Engines

Some search engines will use the name and content attributes of the meta element to index your pages.

The following meta element defines a description of a page:

```
<meta name="description" content="Free Web tutorials on HTML, CSS, XML" />
```

The following meta element defines keywords for a page:

```
<meta name="keywords" content="HTML, CSS, XML" />
```

The intention of the name and content attributes is to describe the content of a page.

Note: A lot of webmasters have used <meta> tags for spamming, like repeating keywords (or using wrong keywords) for higher ranking. Therefore, most search engines have stopped using <meta> tags to index/rank pages.

JavaScript Introduction

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

What You Should Already Know

you should have a basic understanding of the following: **HTML / XHTML**

What is JavaScript?

❖JavaScript was designed to add interactivity to HTML pages

❖JavaScript is a scripting language

❖A scripting language is a lightweight programming language

❖JavaScript is usually embedded directly into HTML pages

❖JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

❖Everyone can use JavaScript without purchasing a license

What can a JavaScript do?

✓**JavaScript can put dynamic text into an HTML page - **A JavaScript statement like this: document.write("<h1>" + name + "</h1>") can write a variable text into an HTML page

✓**JavaScript can react to events - **A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

✓**JavaScript can read and write HTML elements - **A JavaScript can read and change the content of an HTML element

✓**JavaScript can be used to validate data - **A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

✓**JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

✓**JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

Are Java and JavaScript the same?

NO!

Java and JavaScript are two completely different languages in both concept and design!Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

The Real Name is ECMAScript

JavaScript's official name is **ECMAScript.**

ECMAScript is developed and maintained by the **ECMA** organization.

ECMA would be called **European Computer Manufacturers Association**

The language was invented by **Brendan Eich** at **Netscape** (with Navigator 2.0), and has appeared in all Netscape and Microsoft browsers since 1996.

The development of ECMA-262 started in 1996, and the **first edition** of was adopted by the ECMA General Assembly in **June 1997**.The standard was approved as an **international ISO (ISO/IEC 16262) standard in 1998**.

JavaScript

The HTML <script> tag is used to insert a JavaScript into an HTML page.

To insert a JavaScript into an HTML page, we use the <script> tag.

Inside the <script> tag we use the type attribute to define the scripting language.
So, the **<script type="text/javascript">** and **</script>** tells where the JavaScript starts and ends:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Example

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

Where to Put the JavaScript

JavaScripts can be put in **the body** and in **the head sections** of an HTML page.

JavaScripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, or at a later event, such as when a user clicks a button.

Scripts in <head>

Scripts to be executed when they are called, or when an event is triggered, are placed in functions.Put your functions in the head section.

Example

```
<html>
<head>
<script type="text/javascript">
function message( )
{
alert("This alert box was called with the onload event");
}
</script>
</head>
<body onload="message()">
</body>
</html>
```

Scripts in <body>

If you don't want your script to be placed inside a function, or if your script should write page content, it should be placed in the body section.

Example

```
<html>
<head>
</head>
<body>
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>
</html>
```

Scripts in <head> and <body>

You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

Example

```
<html>
<head>
<script type="text/javascript">
function message()
{
alert("This alert box was called with the onload event");
}
</script>
</head>

<body onload="message()">
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>
</html>
```

Using an External JavaScript

If you want to run the same JavaScript on several pages, without having to write the same script on every page, you can write a JavaScript in an external file.

Save the external JavaScript file with a **.js file extension.**

Note: The external script cannot contain the <script></script> tags!

To use the external script, point to the **.js** file in the **"src"** attribute of the **<script>** tag:

Example

```
<html>

<head>

<script type="text/javascript" src="demo.js"></script>

</head>

<body>

</body>

</html>
```

JavaScript Statements

JavaScript is Case Sensitive

JavaScript is a sequence of statements to be executed by the browser.

A JavaScript statement is a command to a browser. The purpose of the command is to tell the browser what to do.

This JavaScript statement tells the browser to write "Hello Sir" to the web page:

document.write("Hello Sir");

It is normal to add a semicolon at the end of each executable statement.

Note: Using semicolons makes it possible to write multiple statements on one line.

JavaScript Code

JavaScript code (or just JavaScript) is a sequence of JavaScript statements.

Each statement is executed by the browser in the sequence they are written.

This example will write a heading and two paragraphs to a web page:

Example

```
<script type="text/javascript">
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Blocks

JavaScript statements can be grouped together in blocks.

Blocks start with a left curly bracket **{**, and ends with a right curly bracket }.

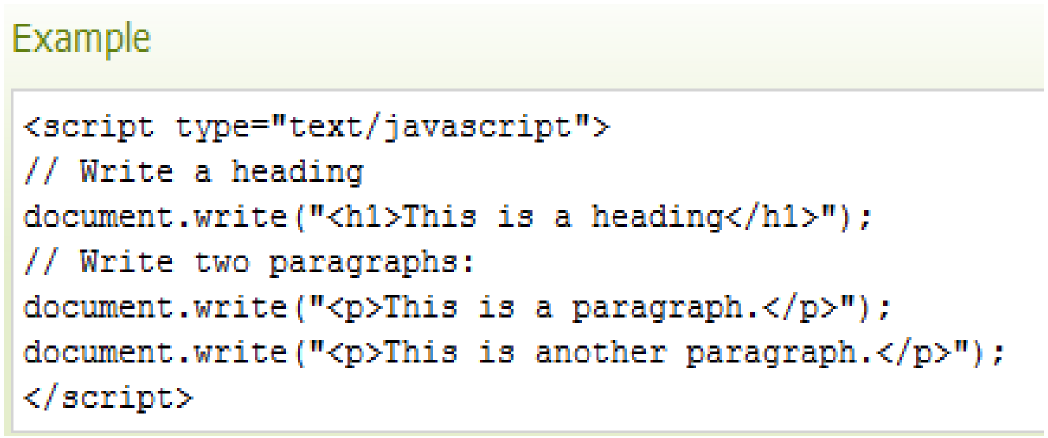
The purpose of a block is to make the sequence of statements execute together.

Normally a block is used to group statements together in a function or in a condition (where a group of statements should be executed if a condition is met).

The HTML comment tag should be used to "hide" the JavaScript.

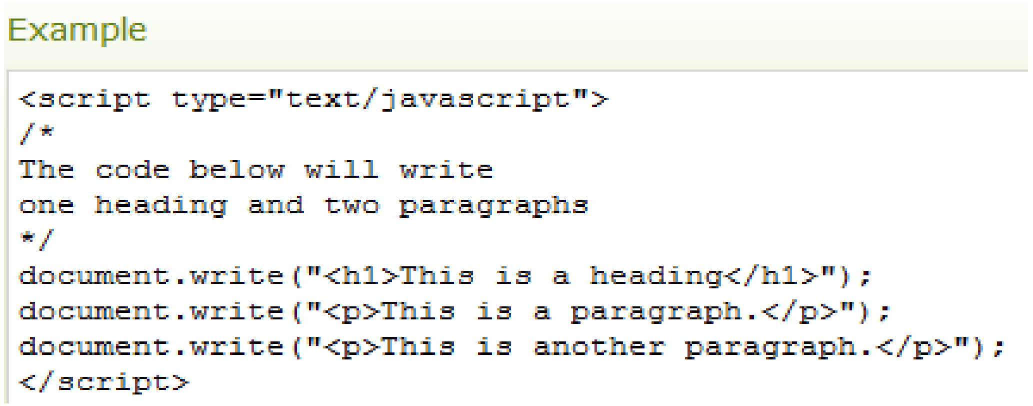
Just add an HTML comment tag **<!--** before the first JavaScript statement, and a **-->** (end of comment) after the last JavaScript statement, like this:

```
<html>

<body>

<script type="text/javascript">

<!--

document.write("Hello World!");

//-->

</script>

</body>

</html>
```



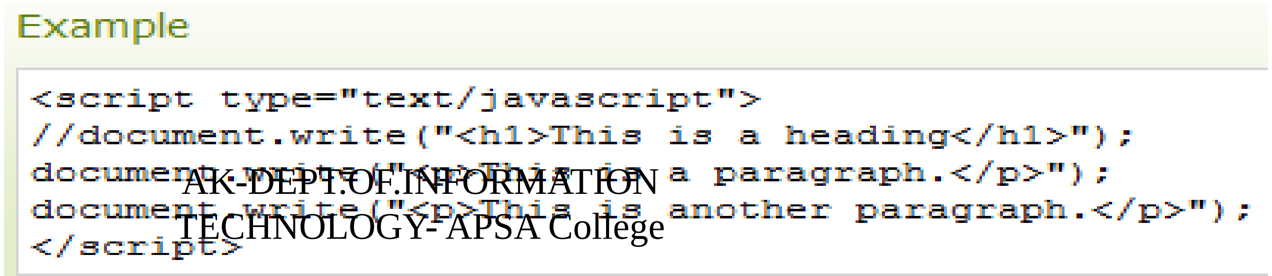
Example

```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs:
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```



Example

```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```



Example

```
<script type="text/javascript">
//document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

JavaScript Variables

Variables are **"containers"** for storing information

As with algebra, JavaScript variables are used to hold values or expressions.

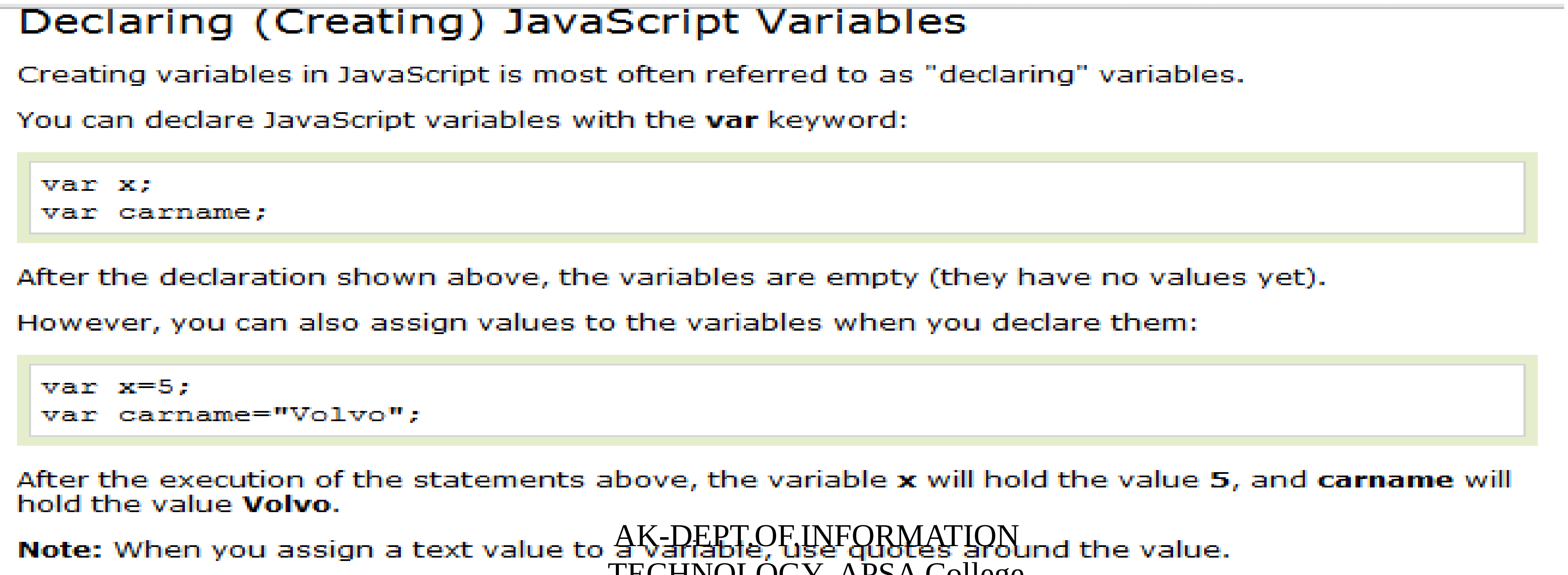
A variable can have a **short name, like x**, or a more **descriptive name, like carname**.

Rules for JavaScript variable names:

•Variable names are case sensitive (y and Y are two different variables)

•Variable names must begin with a letter or the underscore character

Note: Because JavaScript is case-sensitive, variable names are case-sensitive.



Declaring (Creating) JavaScript Variables

Creating variables in JavaScript is most often referred to as "declaring" variables.

You can declare JavaScript variables with the **var** keyword:

```
var x;
var carname;
```

After the declaration shown above, the variables are empty (they have no values yet).

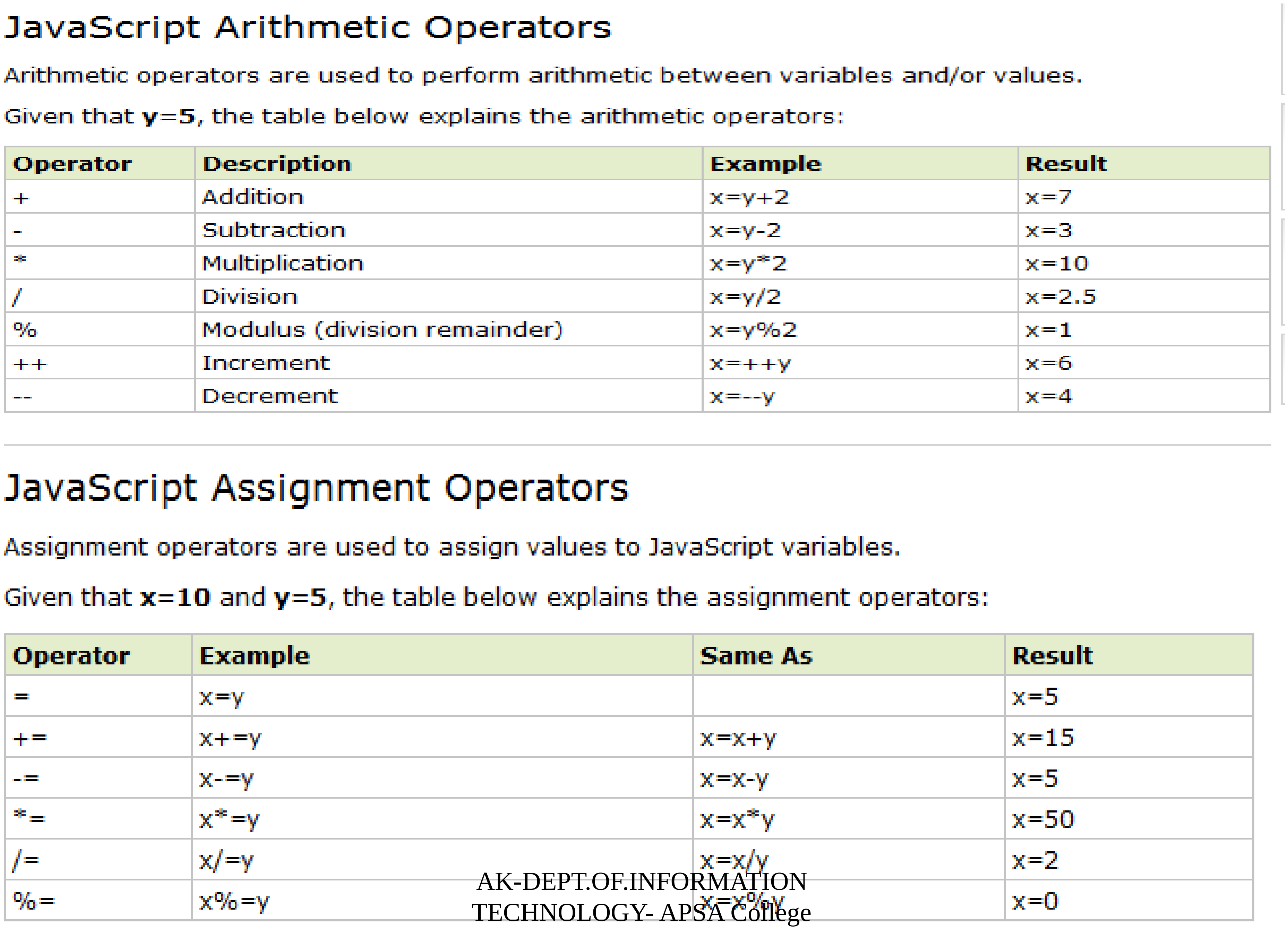
However, you can also assign values to the variables when you declare them:

```
var x=5;
var carname="Volvo";
```

After the execution of the statements above, the variable **x** will hold the value **5**, and **carname** will hold the value **Volvo**.

Note: When you assign a text value to a variable, use quotes around the value.

JavaScript Operators



JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic between variables and/or values.

Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

JavaScript Assignment Operators

Assignment operators are used to assign values to JavaScript variables.

Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operator	Example	Same As	Result
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

The + Operator Used on Strings

The + operator can also be used to add string variables or text values together.

To add two or more string variables together, use the + operator.

```
txt1="What a very";
txt2="nice day";
txt3=txt1+txt2;
```

After the execution of the statements above, the variable txt3 contains "What a verynice day".

To add a space between the two strings, insert a space into one of the strings:

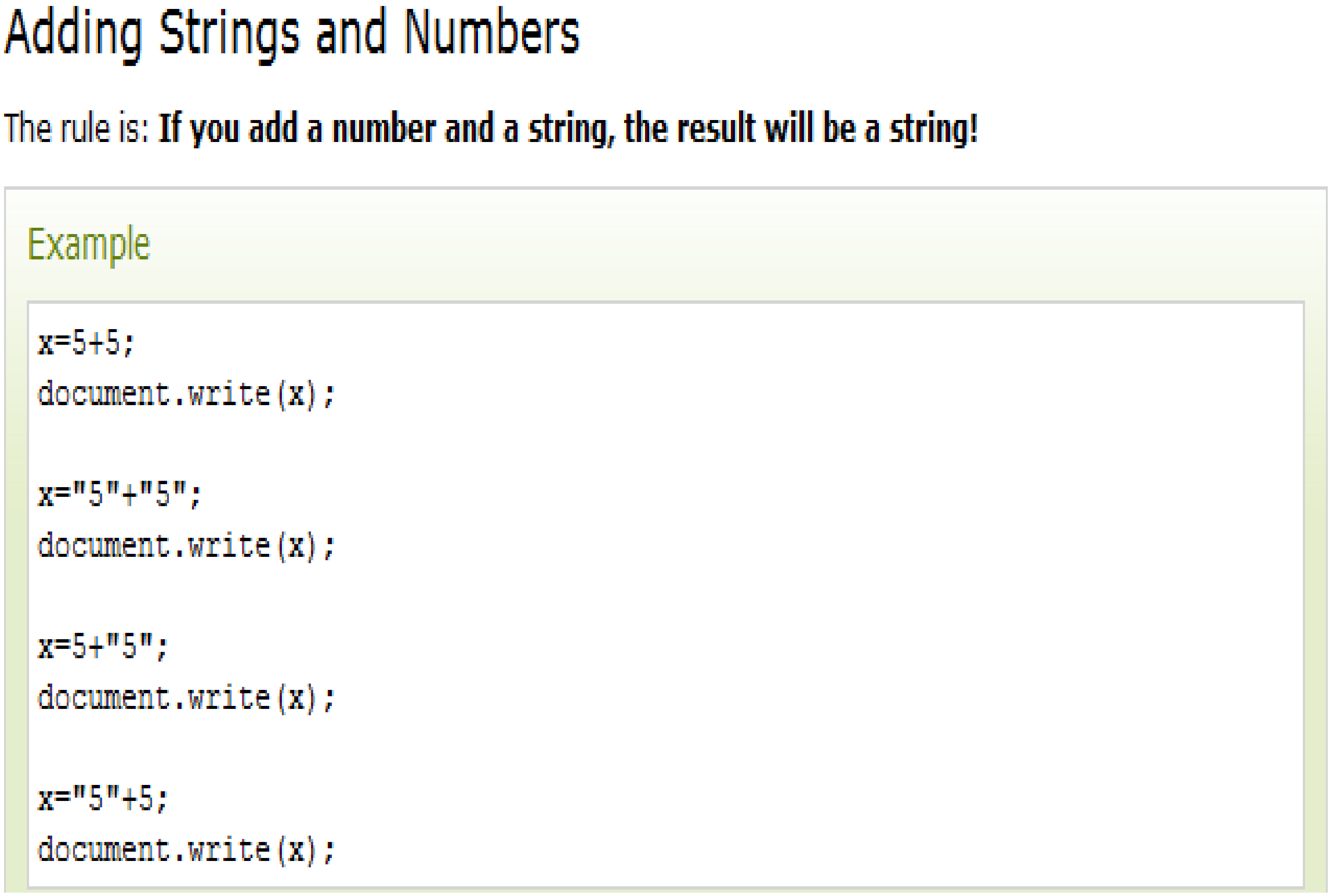
```
txt1="What a very ";
txt2="nice day";
txt3=txt1+txt2;
```

or insert a space into the expression:

```
txt1="What a very";
txt2="nice day";
txt3=txt1+" "+txt2;
```

After the execution of the statements above, the variable txt3 contains:

"What a very nice day"



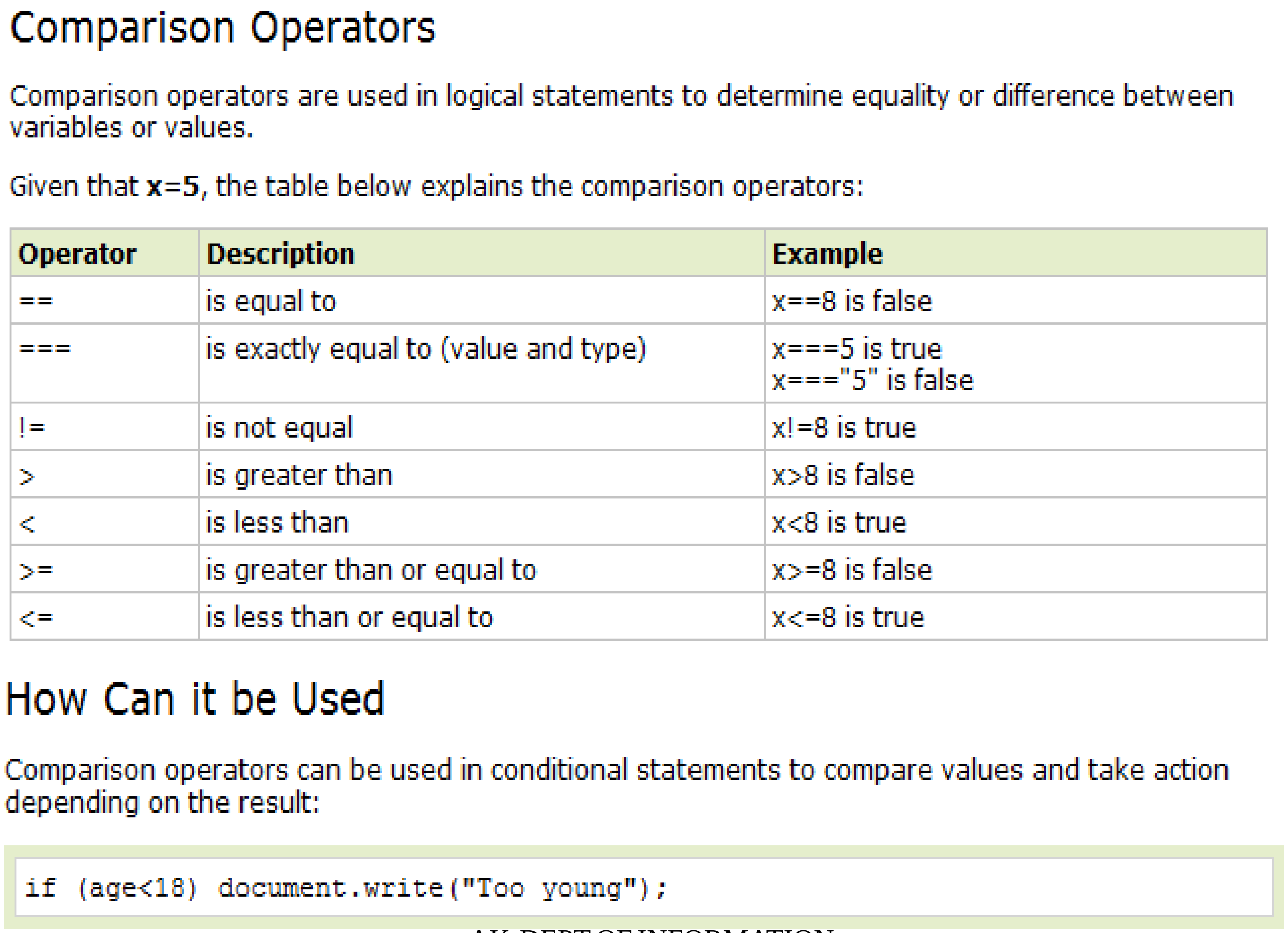
Adding Strings and Numbers

The rule is: **If you add a number and a string, the result will be a string!**

Example

```
x=5+5;
document.write(x);

x="5"+"5";
document.write(x);

x=5+"5";
document.write(x);

x="5"+5;
document.write(x);
```

JavaScript Comparison and Logical Operators



Comparison Operators

Comparison operators are used in logical statements to determine equality or difference between variables or values.

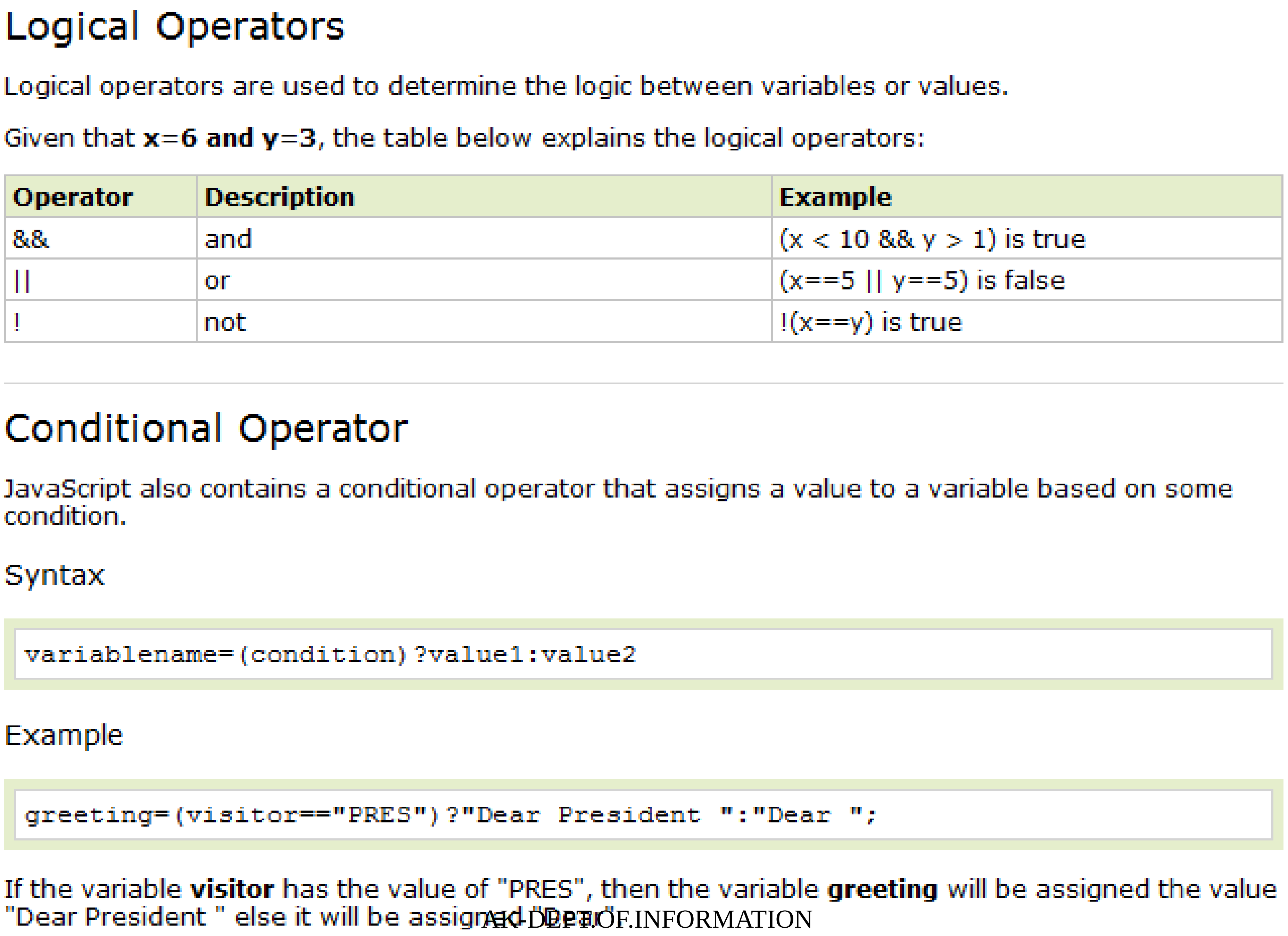
Given that **x=5**, the table below explains the comparison operators:

Operator	**Description**	**Example**
==	is equal to	x==8 is false
===	is exactly equal to (value and type)	x===5 is true x==="5" is false
!=	is not equal	x!=8 is true
>	is greater than	x>8 is false
<	is less than	x<8 is true
>=	is greater than or equal to	x>=8 is false
<=	is less than or equal to	x<=8 is true

How Can it be Used

Comparison operators can be used in conditional statements to compare values and take action depending on the result:

```
if (age<18) document.write("Too young");
```



Logical Operators

Logical operators are used to determine the logic between variables or values.

Given that **x=6 and y=3**, the table below explains the logical operators:

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

Conditional Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax

```
variablename=(condition)?value1:value2
```

Example

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

If the variable **visitor** has the value of "PRES", then the variable **greeting** will be assigned the value "Dear President " else it will be assigned "Dear".

JavaScript Conditional Statements

If Statement

Use the if statement to execute some code only if a specified condition is true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a JavaScript error!

Example

```
<script type="text/javascript">
//Write a "Good morning" greeting if
//the time is less than 10

var d=new Date();
var time=d.getHours();

if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
</script>
```

If...else Statement

Use the if....else statement to execute some code if a condition is true and another code if the condition is not true.

Syntax

```
if (condition)
  {
  code to be executed if condition is true
  }
else
  {
  code to be executed if condition is not true
  }
```

Example

```
<script type="text/javascript">
//If the time is less than 10, you will get a "Good morning" greeting.
//Otherwise you will get a "Good day" greeting.

var d = new Date();
var time = d.getHours();

if (time < 10)
  {
  document.write("Good morning!");
  }
else
  {
  document.write("Good day!");
  }
</script>
```

If...else if...else Statement

Use the if....else if...else statement to select one of several blocks of code to be executed.

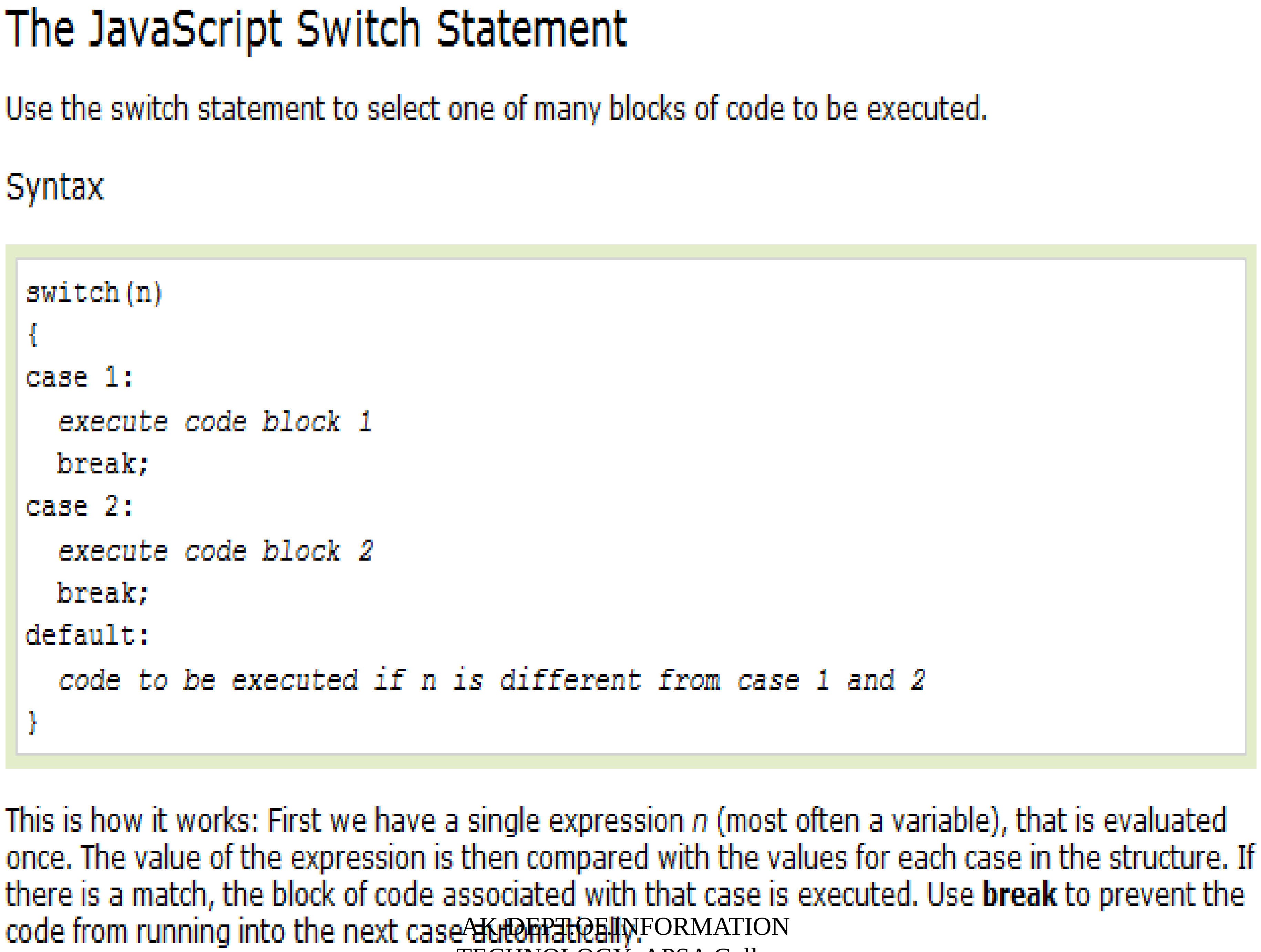
Syntax

```
if (condition1)
  {
  code to be executed if condition1 is true
  }
else if (condition2)
  {
  code to be executed if condition2 is true
  }
else
  {
  code to be executed if condition1 and condition2 are not true
  }
```

Example

```
<script type="text/javascript">
var d = new Date()
var time = d.getHours()
if (time<10)
  {
  document.write("<b>Good morning</b>");
  }
else if (time>10 && time<16)
  {
  document.write("<b>Good day</b>");
  }
else
  {
  document.write("<b>Hello World!</b>");
  }
</script>
```

```
<html>
<body>
<script type="text/javascript">
var r=Math.random();
if (r>0.5)
{
document.write("<a href='http://www.google.com'>Google Search Engine</a>");
}
else
{
document.write("<a href='http://www.yahoo.com'>Yahoo Search Engine</a>");
}
</script>
</body>
</html>
```



The JavaScript Switch Statement

Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

This is how it works: First we have a single expression *n* (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

Example

```
<script type="text/javascript">
//You will receive a different greeting based
//on what day it is. Note that Sunday=0,
//Monday=1, Tuesday=2, etc.

var d=new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("Finally Friday");
  break;
case 6:
  document.write("Super Saturday");
  break;
case 0:
  document.write("Sleepy Sunday");
  break;
default:
  document.write("I'm looking forward to this weekend!");
}
</script>
```

JavaScript Popup Boxes

Alert Box

An alert box is often used if you want to make sure information comes through to the user.

When an alert box pops up, the user will have to click "OK" to proceed.

Syntax

```
alert("sometext");
```

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
alert("I am an alert box!");
}
</script>
</head>
<body>
<input type="button" onclick="show_alert()" value="Show" />
</body>
</html>
```

Confirm Box

A confirm box is often used if you want the user to verify or accept something.

When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

Syntax

```
    confirm("sometext");
```

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Show" />

</body>
</html>
```

Prompt Box

A prompt box is often used if you want the user to input a value before entering a page.

When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.

If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

Syntax

```
prompt("sometext","defaultvalue");
```

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name",“Ananthan");
if (name!=null && name!="")
  {
  document.write("Hello " + name + "! How are you today?");
  }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show
prompt box" />

</body>
</html>
```

JavaScript Functions

To keep the browser from executing a script when the page loads, you can put your script into a function.

A function contains code that will be executed by an event or by a call to the function.

You may call a function from anywhere within a page (or even from other pages if the function is embedded in an external .js file).

Functions can be defined both in the <head> and in the <body> section of a document. However, to assure that a function is read/loaded by the browser before it is called, it could be wise to put functions in the <head> section.

How to Define a Function

Syntax

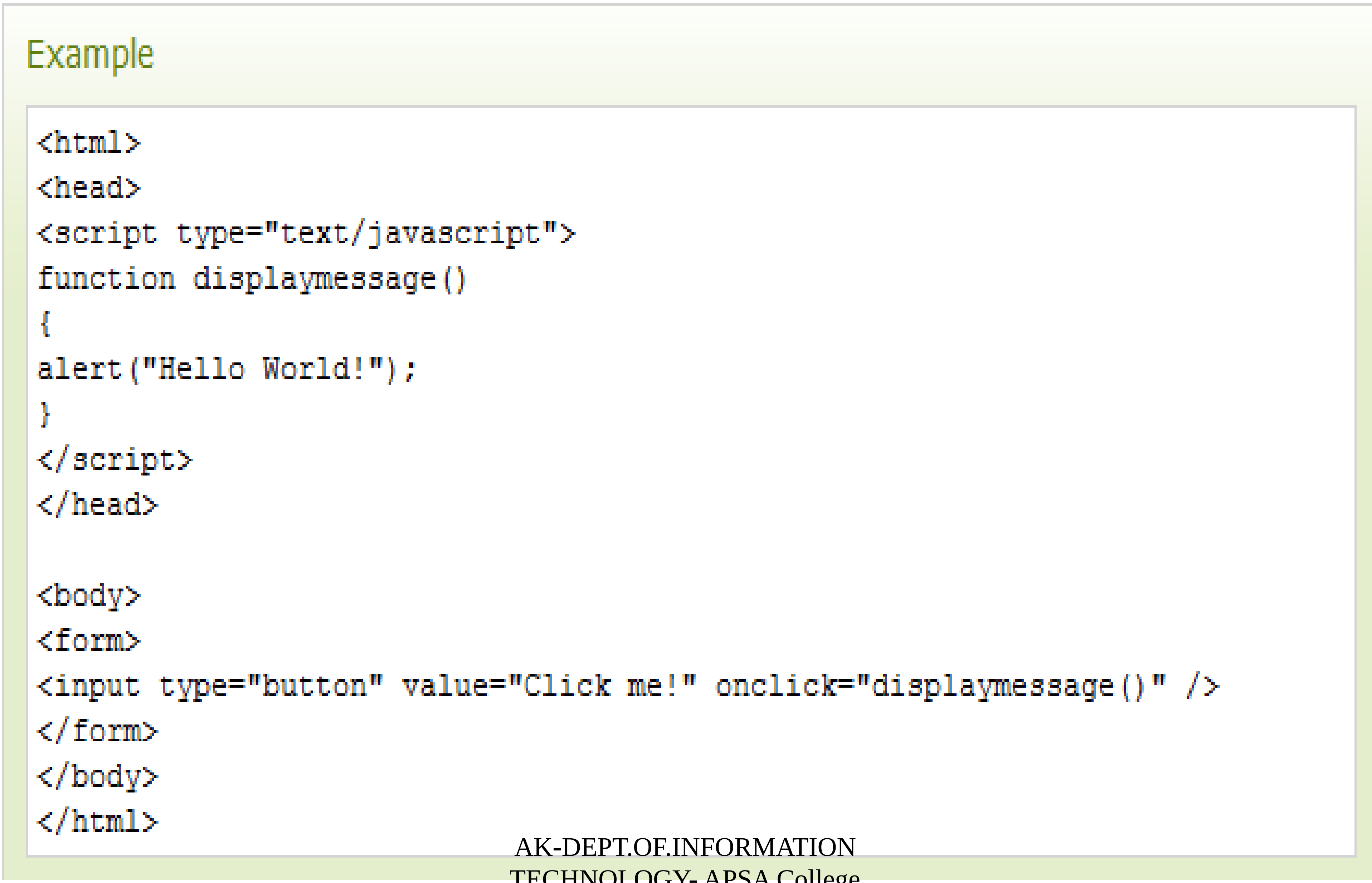
```
function functionname(var1,var2,...,varX)
{
some code
}
```

The parameters var1, var2, etc. are variables or values passed into the function. The { and the } defines the start and end of the function.

Note: A function with no parameters must include the parentheses () after the function name.

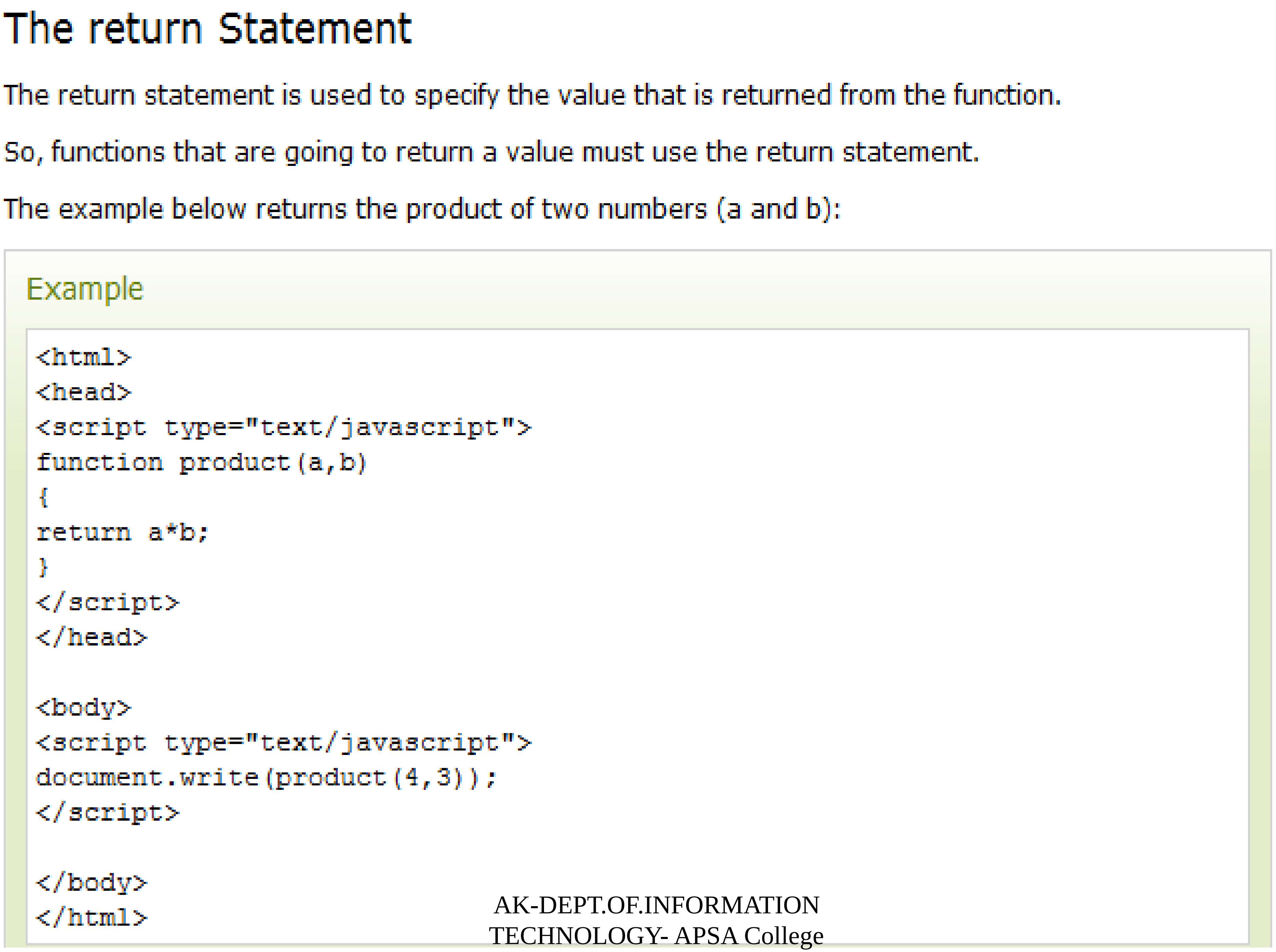
Note: Do not forget about the importance of capitals in JavaScript! The word function must be written in lowercase letters, otherwise a JavaScript error occurs! Also note that you must call a function with the exact same capitals as in the function name.

JavaScript Function Example



Example

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>

<body>
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
</body>
</html>
```



The return Statement

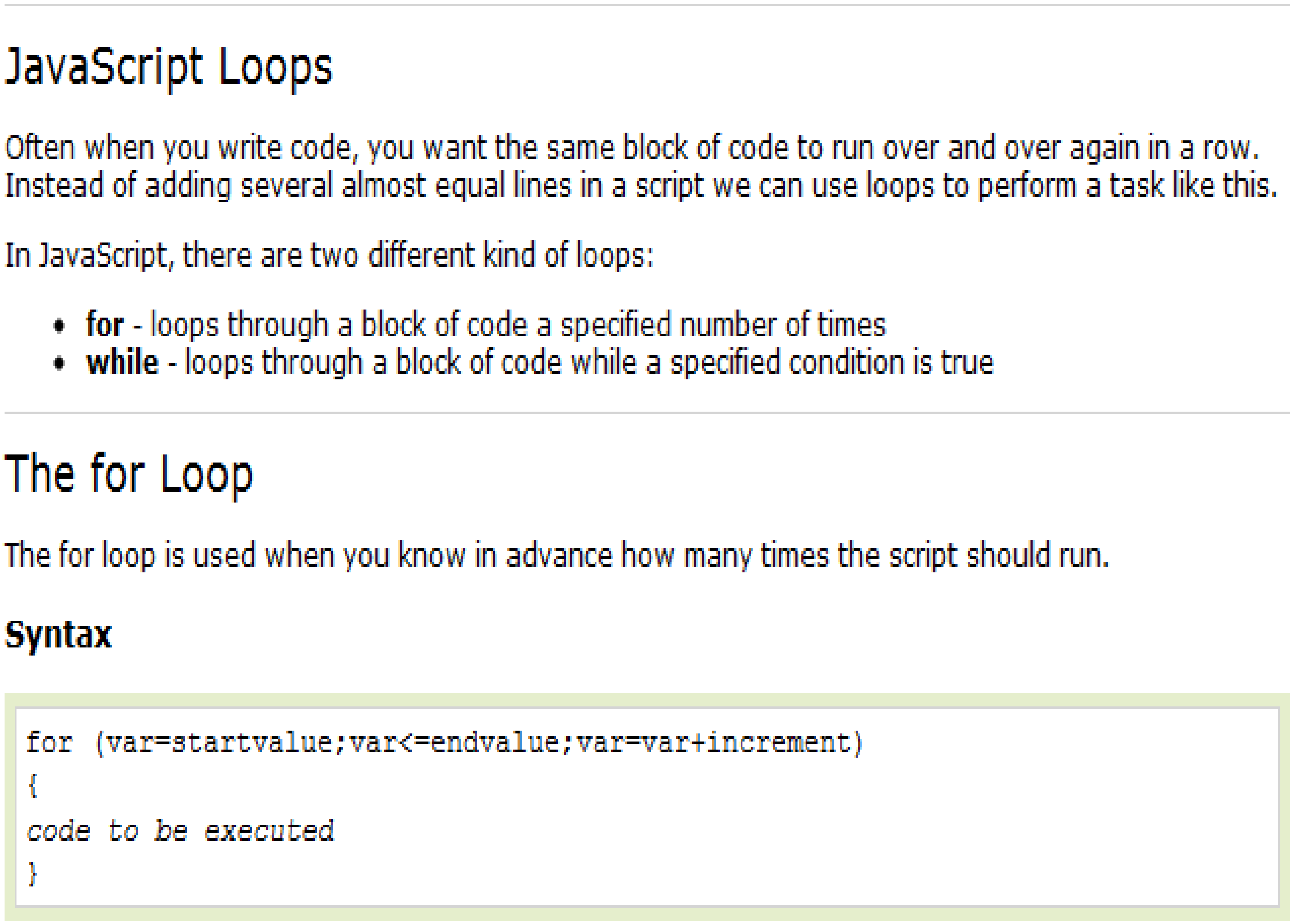
The return statement is used to specify the value that is returned from the function.

So, functions that are going to return a value must use the return statement.

The example below returns the product of two numbers (a and b):

Example

```
<html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>

<body>
<script type="text/javascript">
document.write(product(4,3));
</script>

</body>
</html>
```



JavaScript Loops

Often when you write code, you want the same block of code to run over and over again in a row. Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In JavaScript, there are two different kind of loops:

- **for** - loops through a block of code a specified number of times
- **while** - loops through a block of code while a specified condition is true

The for Loop

The for loop is used when you know in advance how many times the script should run.

Syntax

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

```
<html>
<body>
<script type="text/javascript">
for (i = 1; i <= 6; i++)
{
document.write("<h" + i + ">This is heading " + i);
document.write("</h" + i + ">");
}
</script>
</body>
```

JavaScript For...In Statement

The for...in statement loops through the elements of an array or through the properties of an object.

Syntax

```
for (variable in object)
  {
  code to be executed
  }
```

Note: The code in the body of the for...in loop is executed once for each element/property.

Note: The variable argument can be a named variable, an array element, or a property of an object.

Example

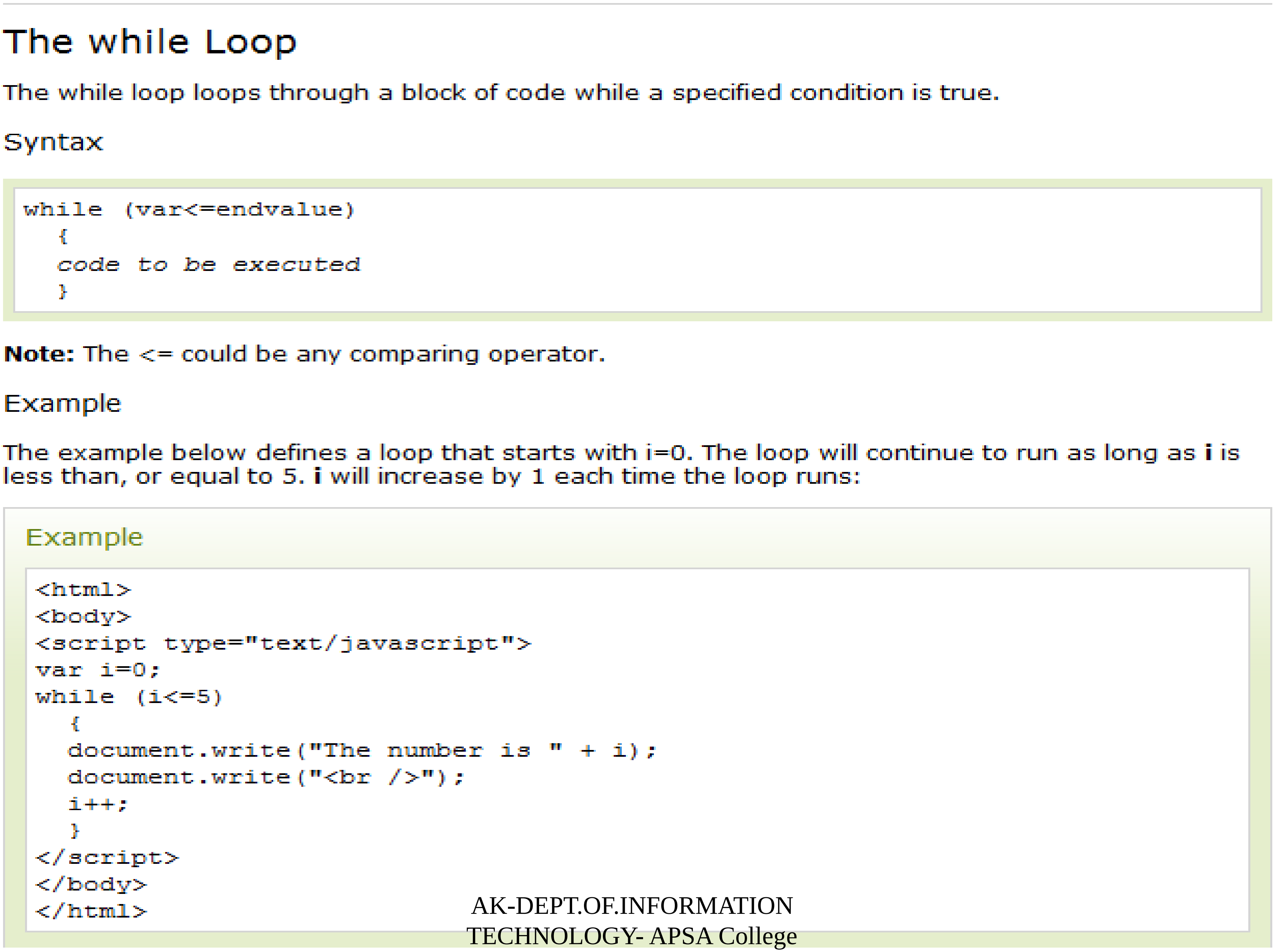
Use the for...in statement to loop through an array:

Example

```
<html>
<body>

<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";

for (x in mycars)
  {
  document.write(mycars[x] + "<br />");
  }
</script>

</body>
</html>
```

Here is the following example that prints out the properties of a Web browser's Navigator object:

```
<script type="text/javascript">
var aProperty;
document.write("Navigator Object Properties<br /> ");
for (aProperty in navigator)
{
  document.write(aProperty);
  document.write("<br />");
}
document.write("Exiting from the loop!");
</script>
```



The while Loop

The while loop loops through a block of code while a specified condition is true.

Syntax

```
while (var<=endvalue)
  {
  code to be executed
  }
```

Note: The <= could be any comparing operator.

Example

The example below defines a loop that starts with i=0. The loop will continue to run as long as **i** is less than, or equal to 5. **i** will increase by 1 each time the loop runs:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
while (i<=5)
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
</script>
</body>
</html>
```

The do...while Loop

The do...while loop is a variant of the while loop. This loop will execute the block of code ONCE, and then it will repeat the loop as long as the specified condition is true.

Syntax

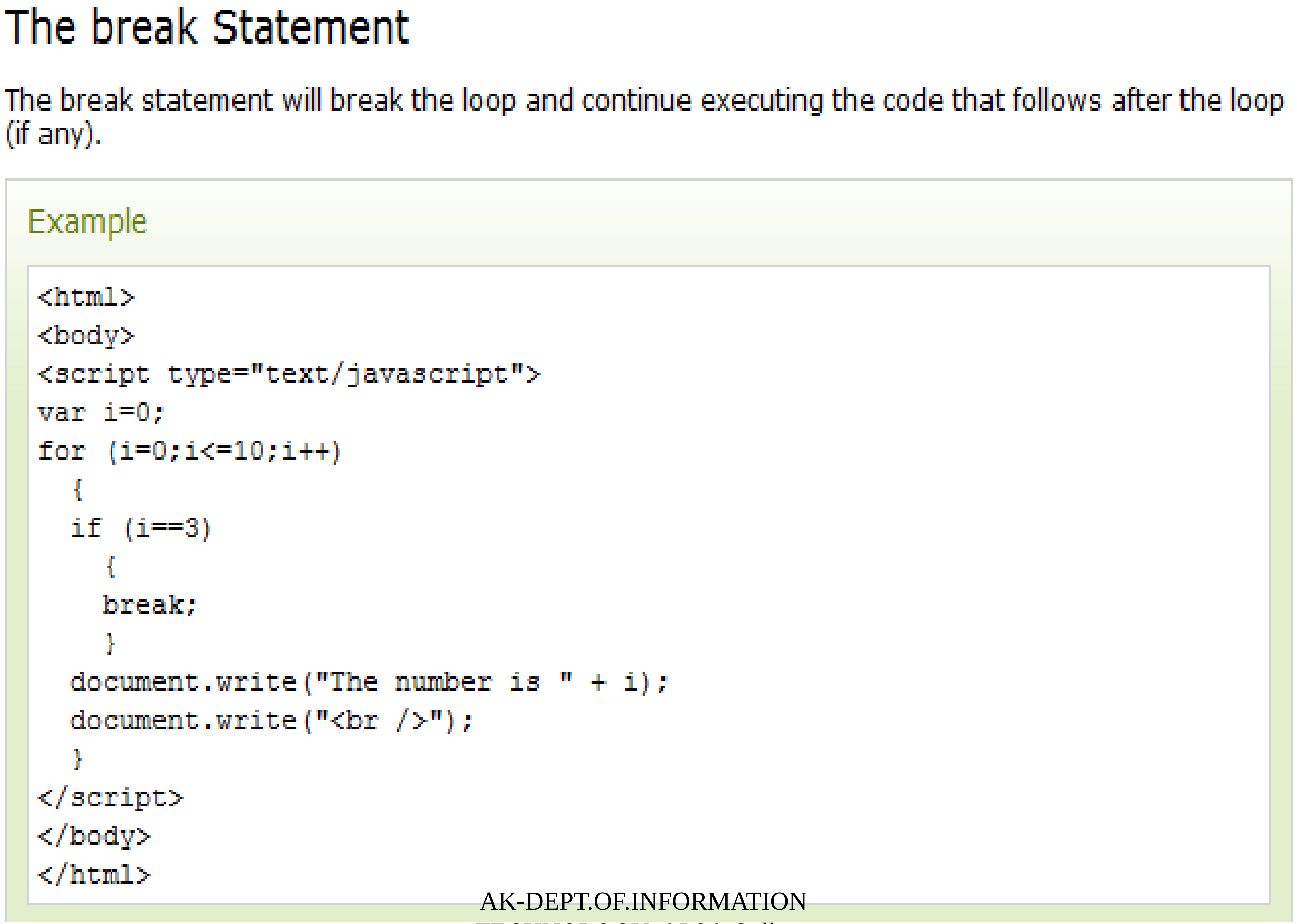
```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

Example

The example below uses a do...while loop. The do...while loop will always be executed at least once, even if the condition is false, because the statements are executed before the condition is tested:

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
do
  {
  document.write("The number is " + i);
  document.write("<br />");
  i++;
  }
while (i<=5);
</script>
</body>
</html>
```



The break Statement

The break statement will break the loop and continue executing the code that follows after the loop (if any).

Example

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    break;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

The continue Statement

The continue statement will break the current loop and continue with the next value.

Example

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
  if (i==3)
    {
    continue;
    }
  document.write("The number is " + i);
  document.write("<br />");
  }
</script>
</body>
</html>
```

```
<script type="text/javascript">
<!--
document.write("Entering the loop!<br /> ");
outerloop:
for (var i = 0; i < 5; i++)
{
  document.write("Outerloop: " + i + "<br />");
  innerloop:
  for (var j = 0; j < 5; j++)
  {
     if (j >  3 ) break ;
     if (i == 2) break innerloop;
     if (i == 4) break outerloop;
     document.write("Innerloop: " + j + "  <br />");
   }
}
document.write("Exiting the loop!<br /> ");
//-->
</script>
```

The **Array** object let's you store multiple values in a single variable.

Syntax:

Creating a **Array** object:

```
var fruits = new Array( "apple", "orange", "mango" );
```

The *Array* parameter is a list of strings or integers. When you specify a single numeric parameter with the Array constructor, you specify the initial length of the array. The maximum length allowed for an array is 4,294,967,295.

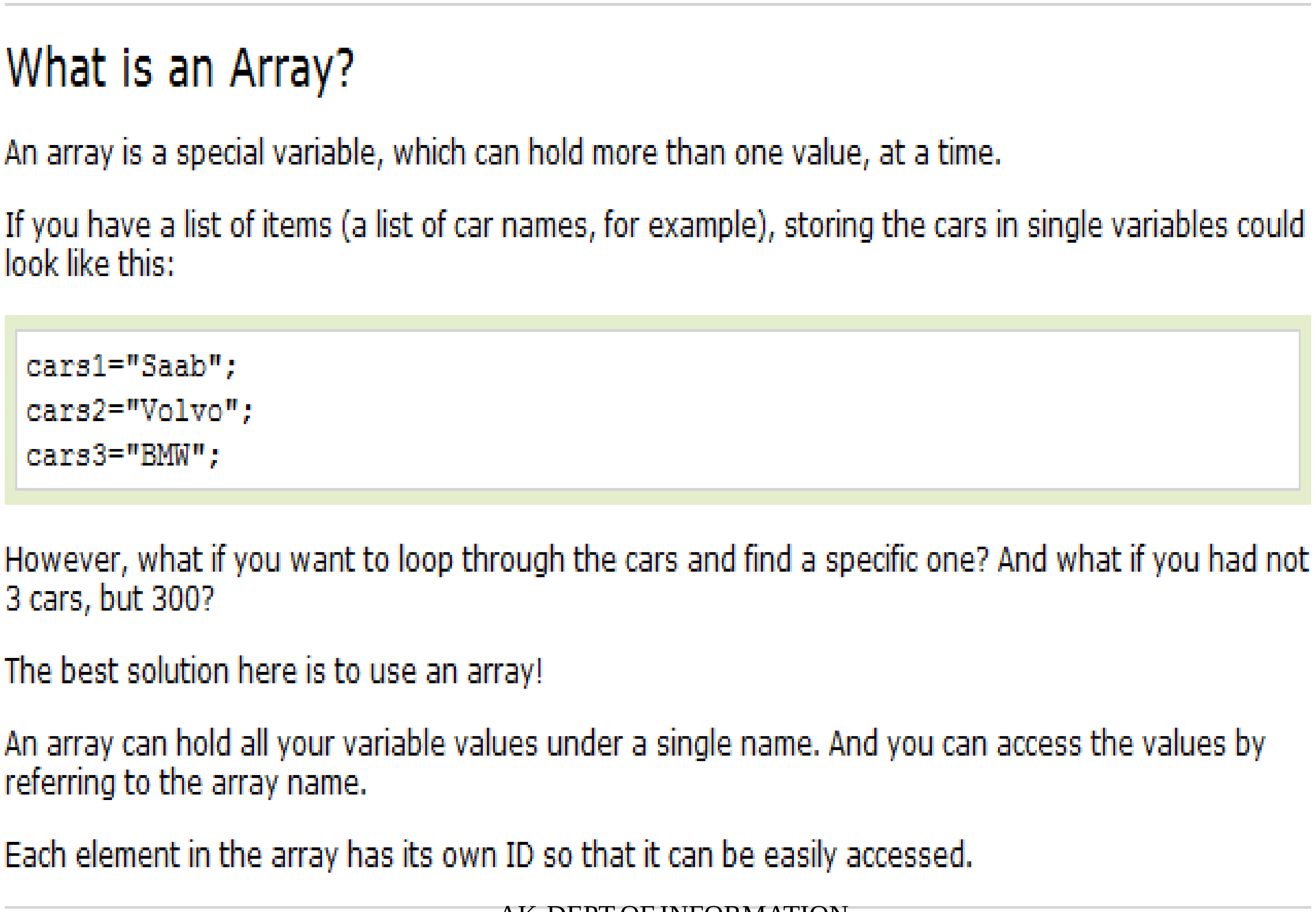
You can create array by simply assigning values as follows:

```
var fruits = [ "apple", "orange", "mango" ];
```

You will use ordinal numbers to access and to set values inside an array as follows:

- fruits[0] is the first element
- fruits[1] is the second element
- fruits[2] is the third element

JavaScript Array Object



What is an Array?

An array is a special variable, which can hold more than one value, at a time.

If you have a list of items (a list of car names, for example), storing the cars in single variables could look like this:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

However, what if you want to loop through the cars and find a specific one? And what if you had not 3 cars, but 300?

The best solution here is to use an array!

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own ID so that it can be easily accessed.

Create an Array

An array can be defined in three ways.

The following code creates an Array object called myCars:

1:

```
var myCars=new Array(); // regular array (add an optional integer
myCars[0]="Saab";       // argument to control array's size)
myCars[1]="Volvo";
myCars[2]="BMW";
```

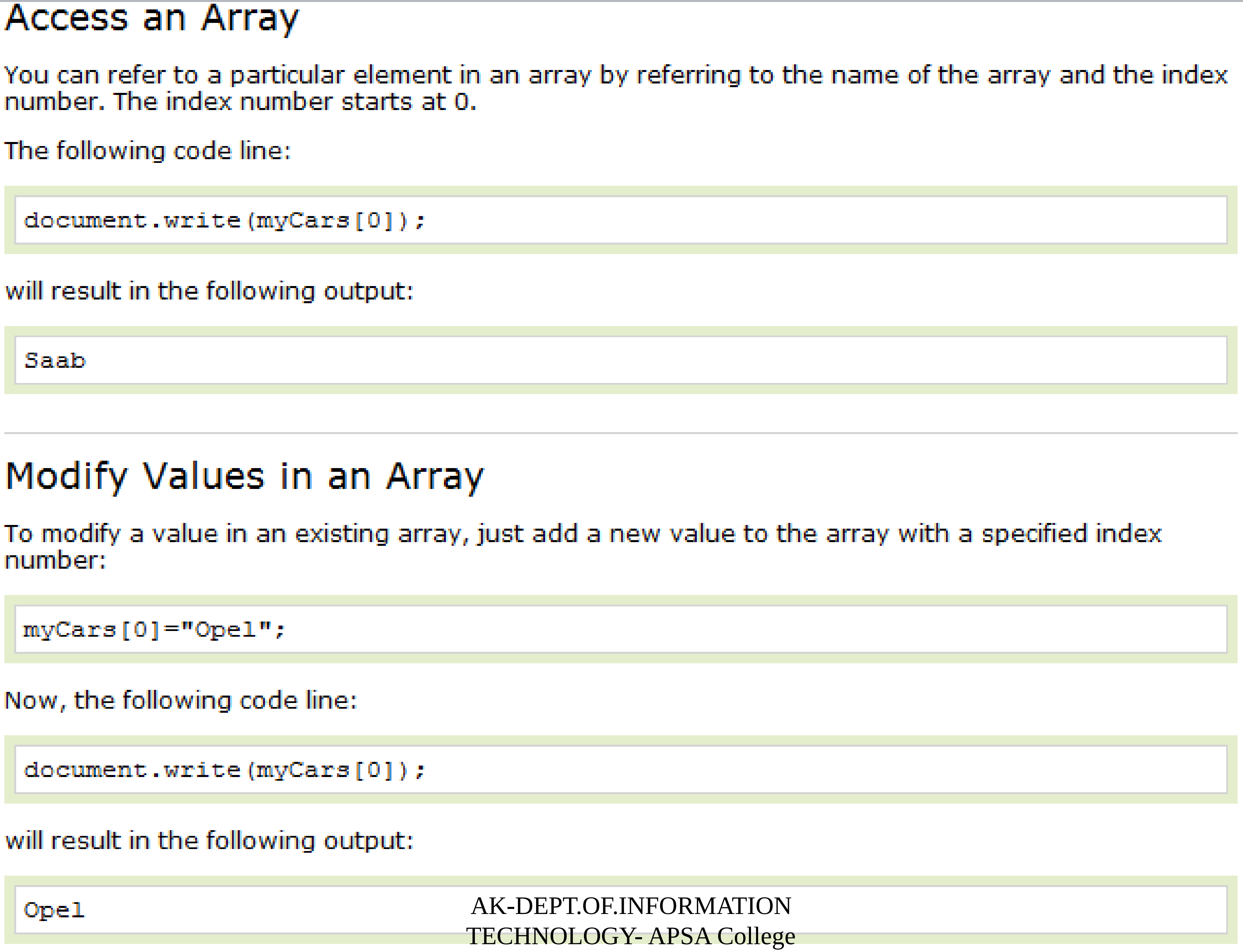
2:

```
var myCars=new Array("Saab","Volvo","BMW"); // condensed array
```

3:

```
var myCars=["Saab","Volvo","BMW"]; // literal array
```

Note: If you specify numbers or true/false values inside the array then the variable type will be Number or Boolean, instead of String.



Access an Array

You can refer to a particular element in an array by referring to the name of the array and the index number. The index number starts at 0.

The following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Saab
```

Modify Values in an Array

To modify a value in an existing array, just add a new value to the array with a specified index number:

```
myCars[0]="Opel";
```

Now, the following code line:

```
document.write(myCars[0]);
```

will result in the following output:

```
Opel
```

Array Object Properties

Property	Description
constructor	Returns the function that created the Array object's prototype
length	Sets or returns the number of elements in an array
prototype	Allows you to add properties and methods to an object

Array Object Methods

Method	Description
concat()	Joins two or more arrays, and returns a copy of the joined arrays
join()	Joins all elements of an array into a string
pop()	Removes the last element of an array, and returns that element
push()	Adds new elements to the end of an array, and returns the new length
reverse()	Reverses the order of the elements in an array
shift()	Removes the first element of an array, and returns that element
slice()	Selects a part of an array, and returns the new array
sort()	Sorts the elements of an array
splice()	Adds/Removes elements from an array
toString()	Converts an array to a string, and returns the result
unshift()	Adds new elements to the beginning of an array, and returns the new length
valueOf()	Returns the primitive value of an array

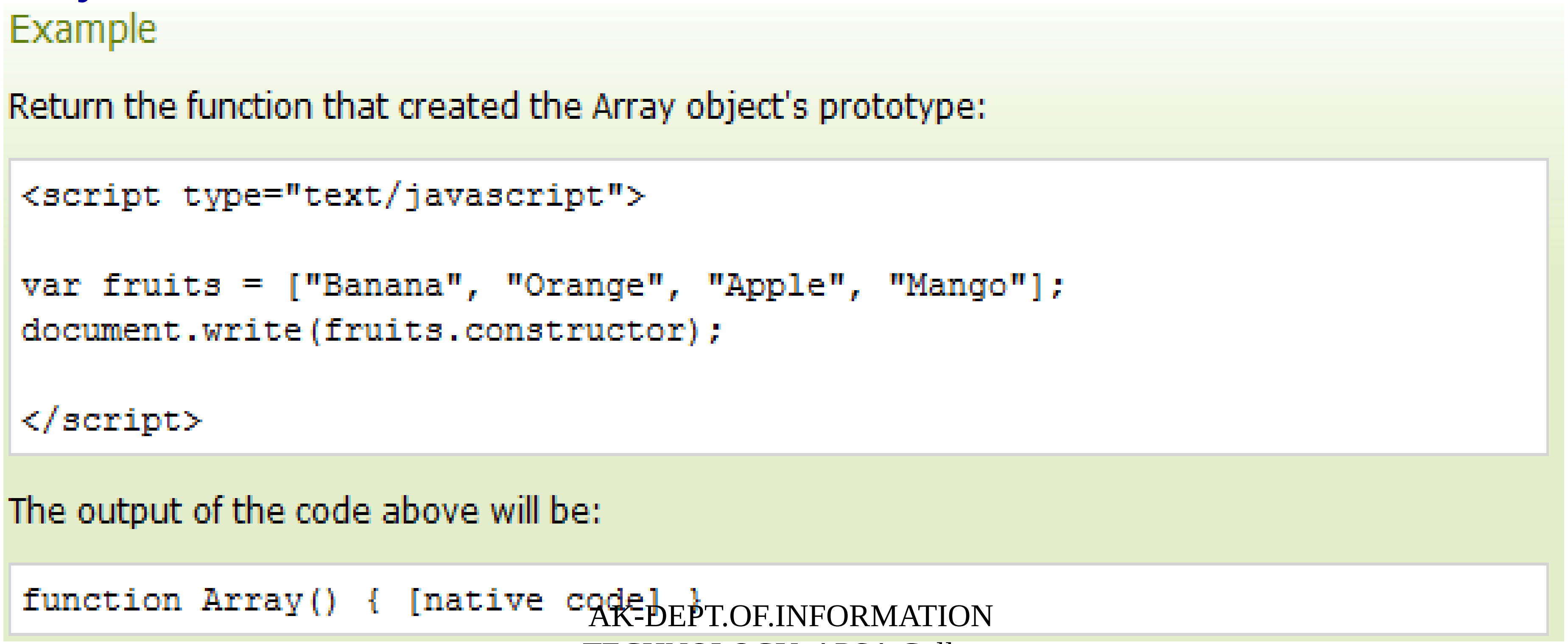
JavaScript constructor Property

Definition and Usage

The constructor property returns the function that created the array object's prototype.

Syntax

array.constructor



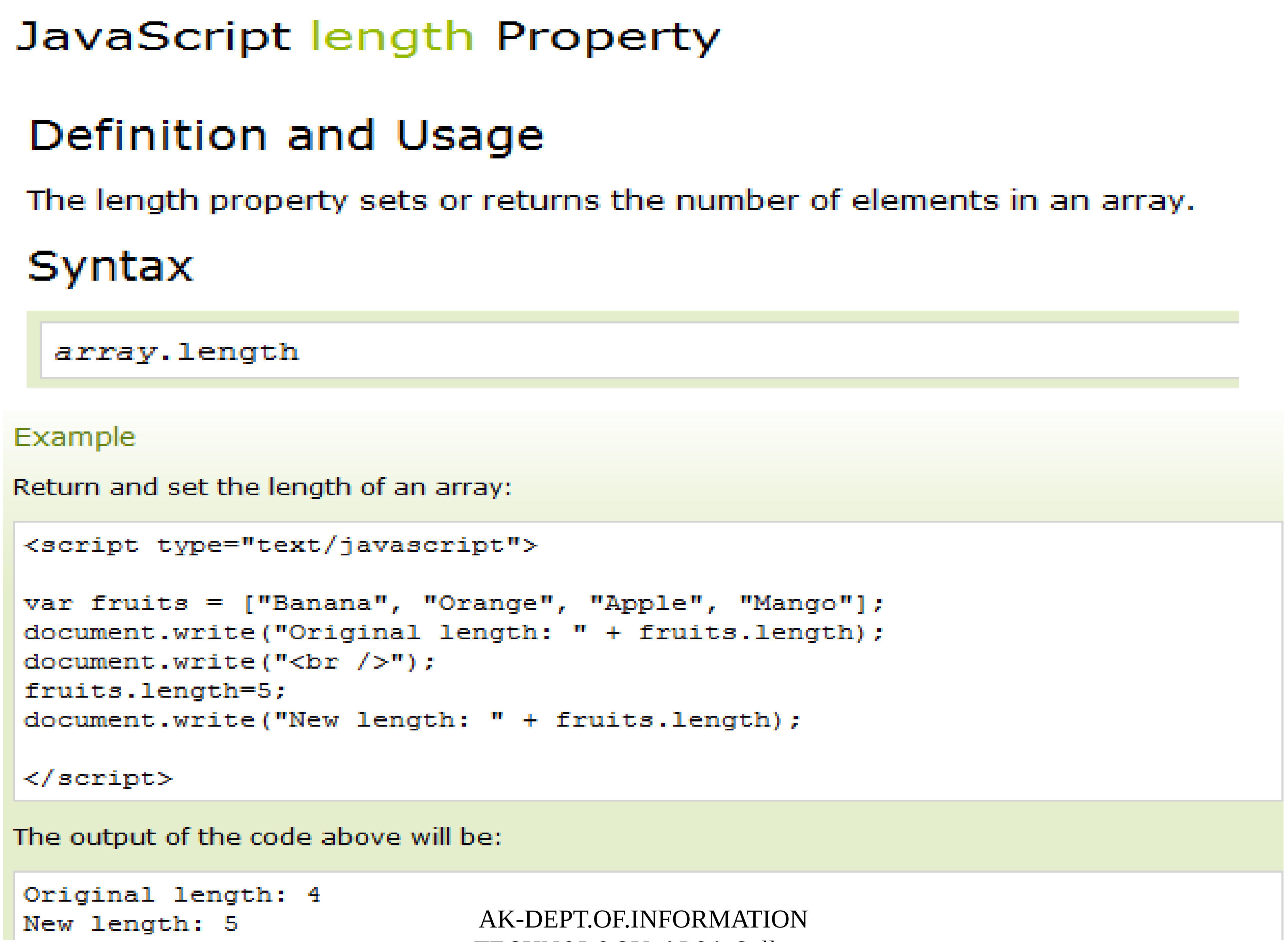
Example

Return the function that created the Array object's prototype:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.constructor);

</script>
```

The output of the code above will be:

```
function Array() { [native code] }
```



JavaScript length Property

Definition and Usage

The length property sets or returns the number of elements in an array.

Syntax

```
array.length
```

Example

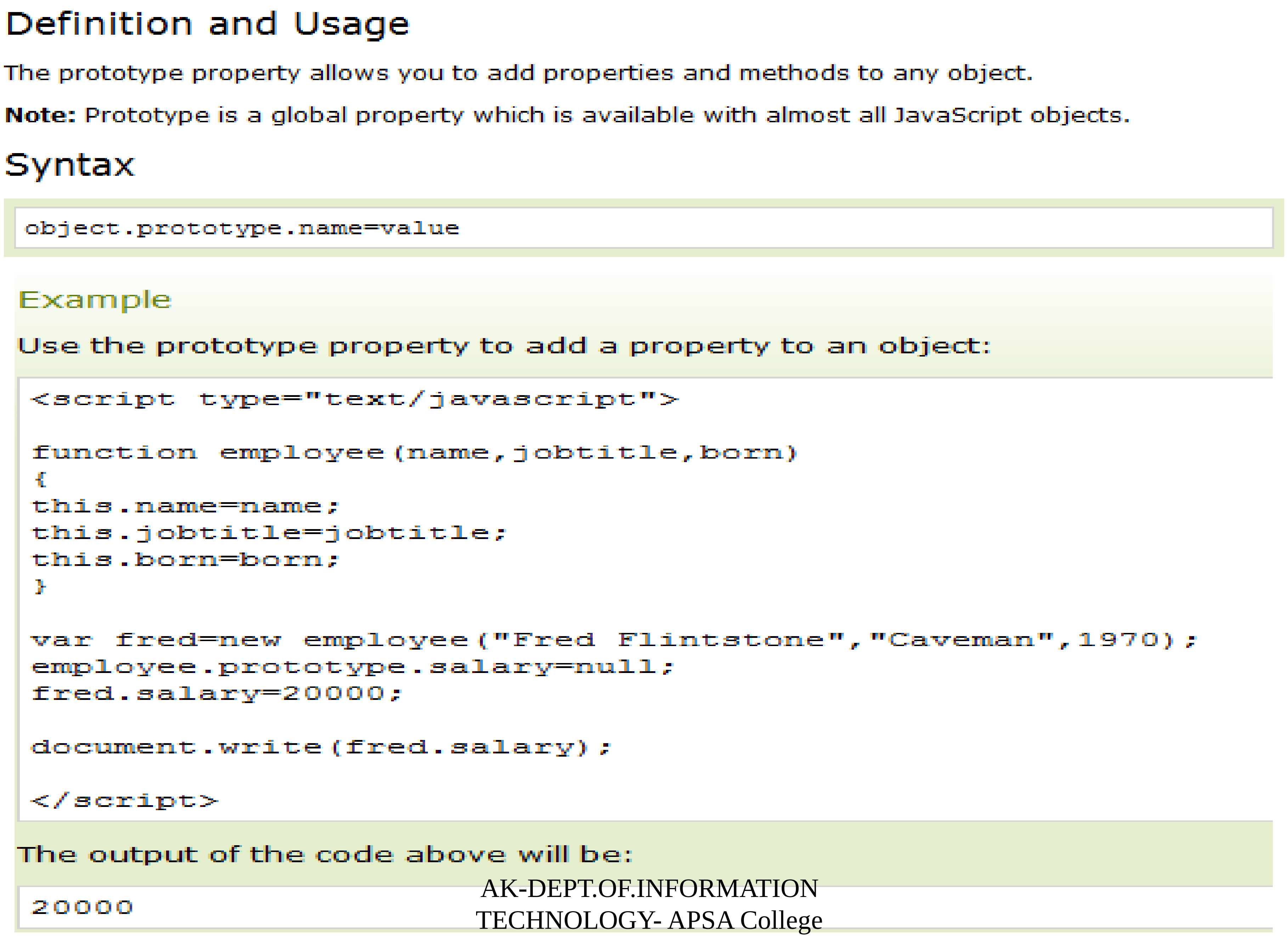
Return and set the length of an array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Original length: " + fruits.length);
document.write("<br />");
fruits.length=5;
document.write("New length: " + fruits.length);

</script>
```

The output of the code above will be:

```
Original length: 4
New length: 5
```

JavaScript prototype Property



Definition and Usage

The prototype property allows you to add properties and methods to any object.

Note: Prototype is a global property which is available with almost all JavaScript objects.

Syntax

```
object.prototype.name=value
```

Example

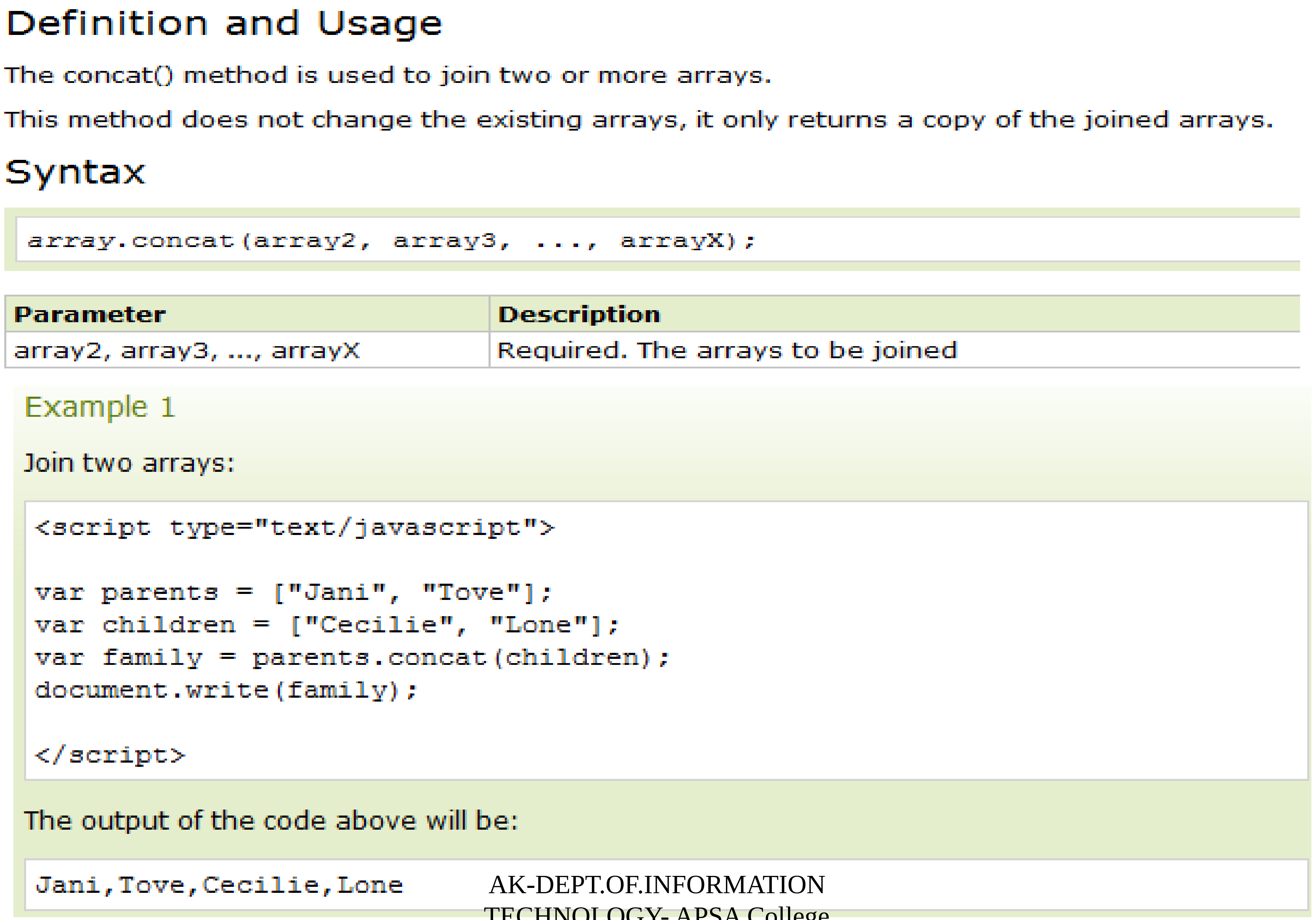
Use the prototype property to add a property to an object:

```
<script type="text/javascript">

function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}

var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;

document.write(fred.salary);

</script>
```

The output of the code above will be:

```
20000
```

JavaScript concat() Method



Definition and Usage

The concat() method is used to join two or more arrays.

This method does not change the existing arrays, it only returns a copy of the joined arrays.

Syntax

```
array.concat(array2, array3, ..., arrayX);
```

Parameter	Description
array2, array3, ..., arrayX	Required. The arrays to be joined

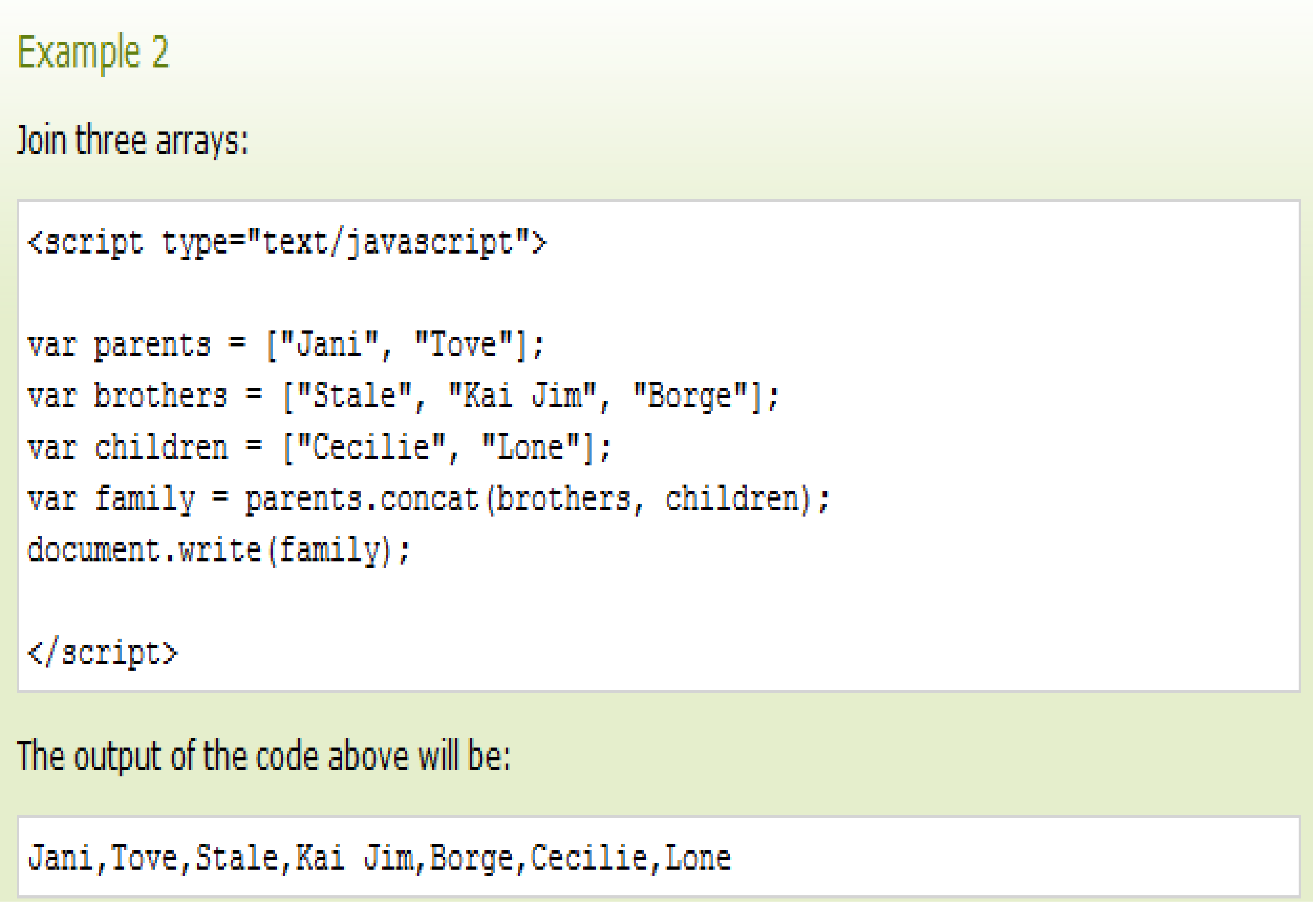
Example 1

Join two arrays:

```
<script type="text/javascript">

var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);

</script>
```

The output of the code above will be:

```
Jani,Tove,Cecilie,Lone
```



Example 2

Join three arrays:

```
<script type="text/javascript">

var parents = ["Jani", "Tove"];
var brothers = ["Stale", "Kai Jim", "Borge"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(brothers, children);
document.write(family);

</script>
```

The output of the code above will be:

```
Jani,Tove,Stale,Kai Jim,Borge,Cecilie,Lone
```

JavaScript join() Method

Definition and Usage

The join() method joins all elements of an array into a string, and returns the string.

The elements will be separated by a specified separator. The default separator is comma (,).

Syntax

```
array.join(separator)
```

Parameter	Description
separator	Optional. The separator to be used. If omitted, the elements are separated with a comma

Example

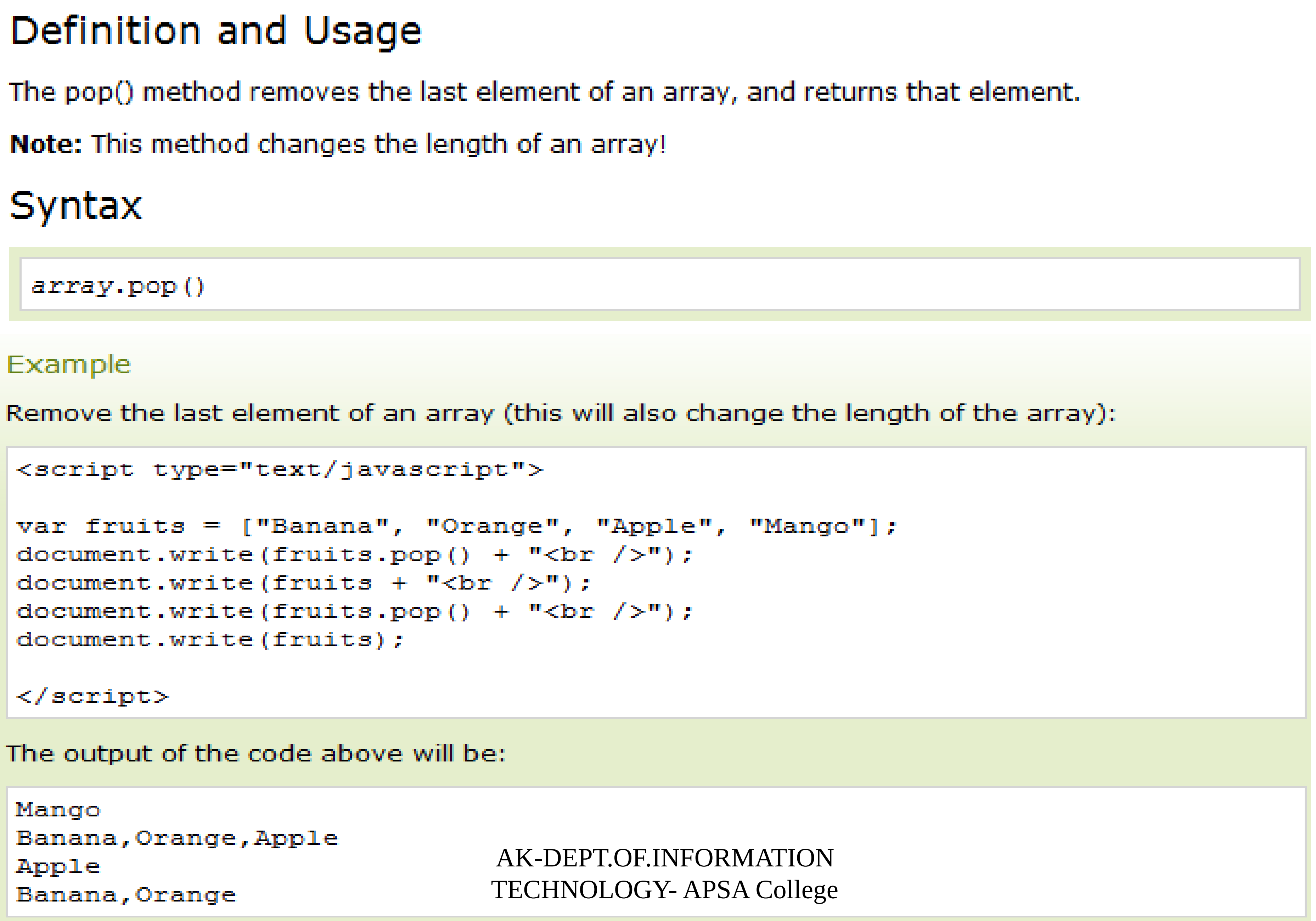
Join all elements of an array into a string:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));

</script>
```

The output of the code above will be:

```
Banana,Orange,Apple,Mango
Banana+Orange+Apple+Mango
Banana and Orange and Apple and Mango
```

JavaScript pop() Method



Definition and Usage

The pop() method removes the last element of an array, and returns that element.

Note: This method changes the length of an array!

Syntax

```
array.pop()
```

Example

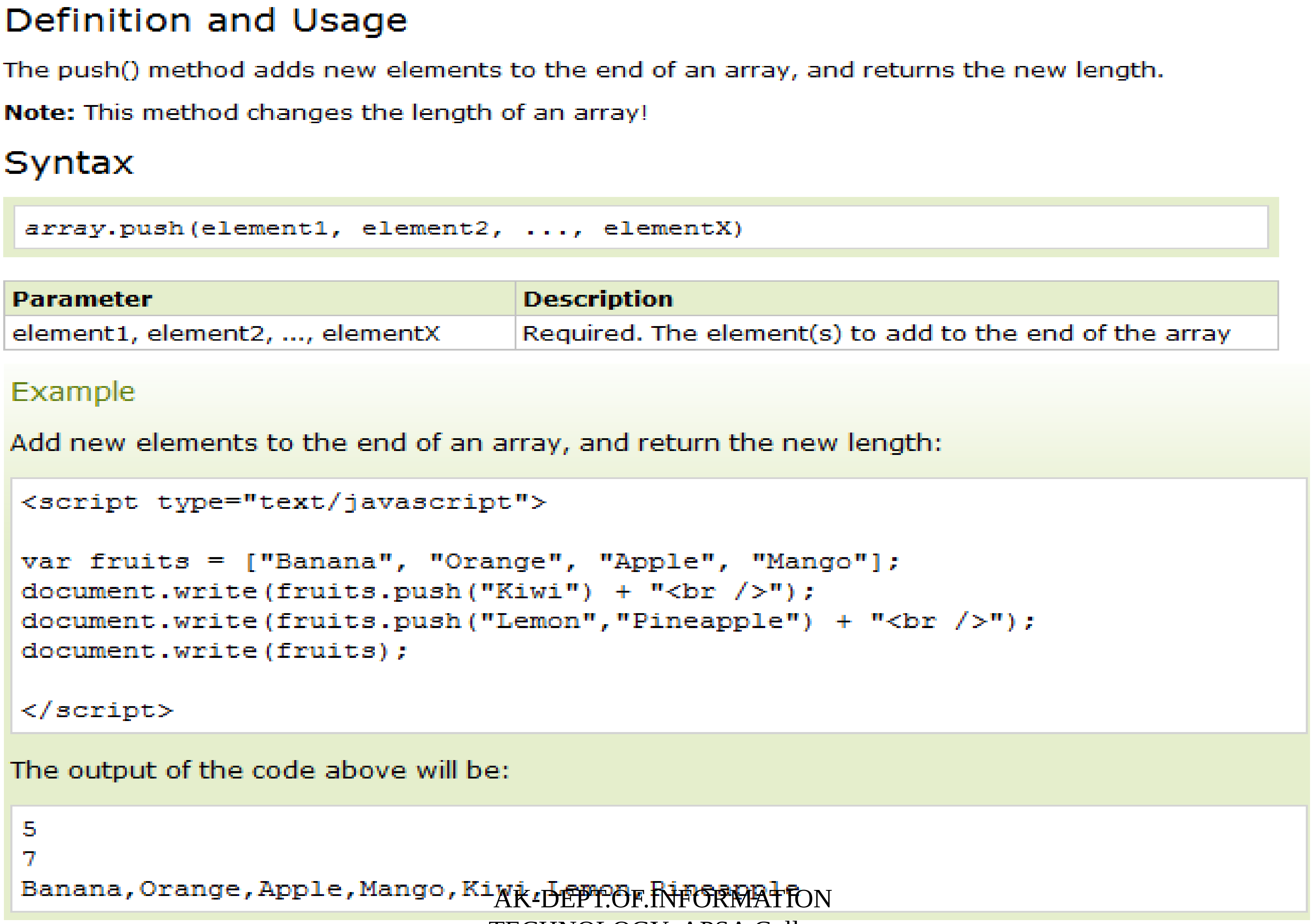
Remove the last element of an array (this will also change the length of the array):

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.pop() + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Mango
Banana,Orange,Apple
Apple
Banana,Orange
```

JavaScript push() Method



Definition and Usage

The push() method adds new elements to the end of an array, and returns the new length.

Note: This method changes the length of an array!

Syntax

```
array.push(element1, element2, ..., elementX)
```

Parameter	Description
element1, element2, ..., elementX	Required. The element(s) to add to the end of the array

Example

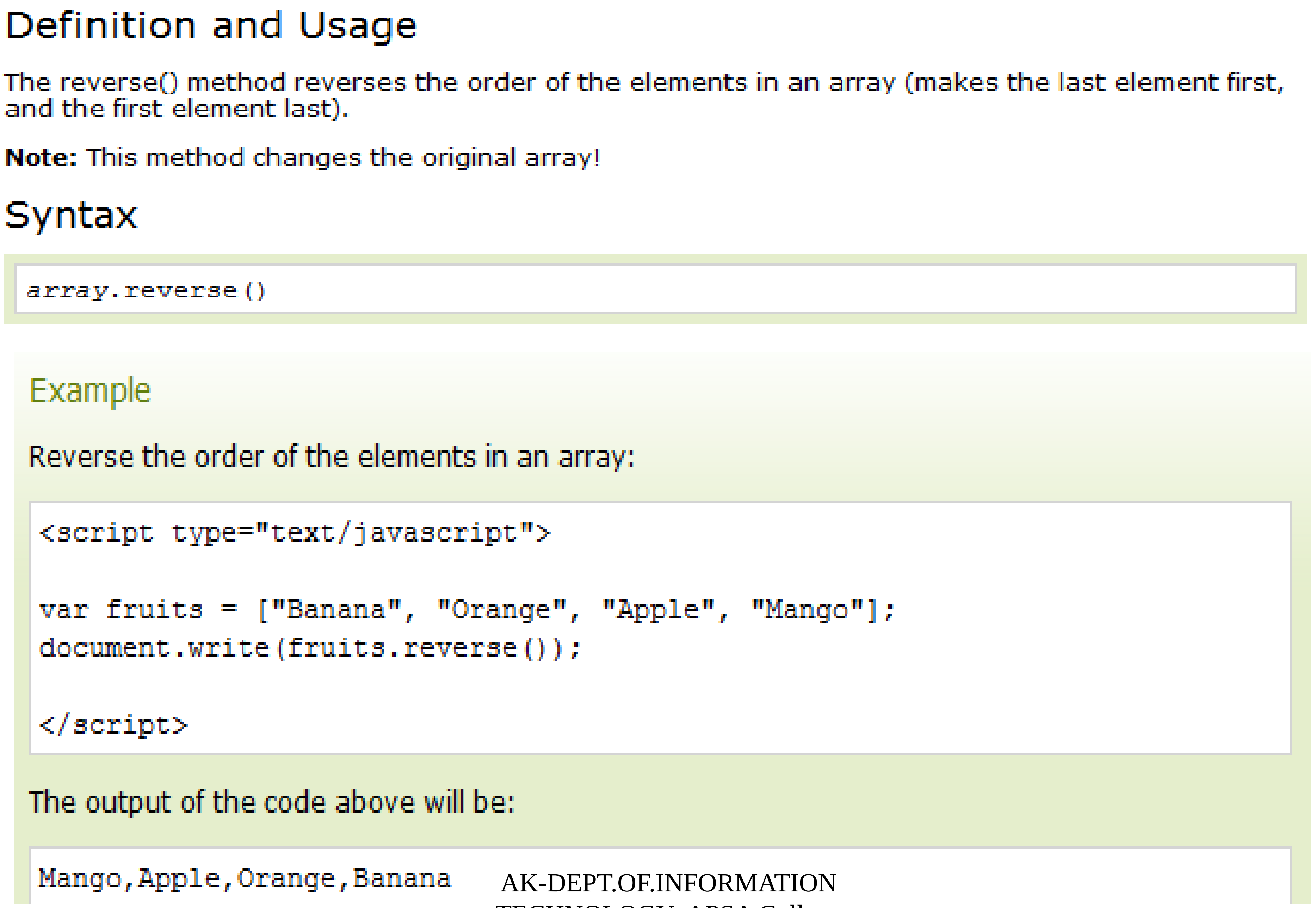
Add new elements to the end of an array, and return the new length:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("Kiwi") + "<br />");
document.write(fruits.push("Lemon","Pineapple") + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
5
7
Banana,Orange,Apple,Mango,Kiwi,Lemon,Pineapple
```

JavaScript reverse() Method



Definition and Usage

The reverse() method reverses the order of the elements in an array (makes the last element first, and the first element last).

Note: This method changes the original array!

Syntax

```
array.reverse()
```

Example

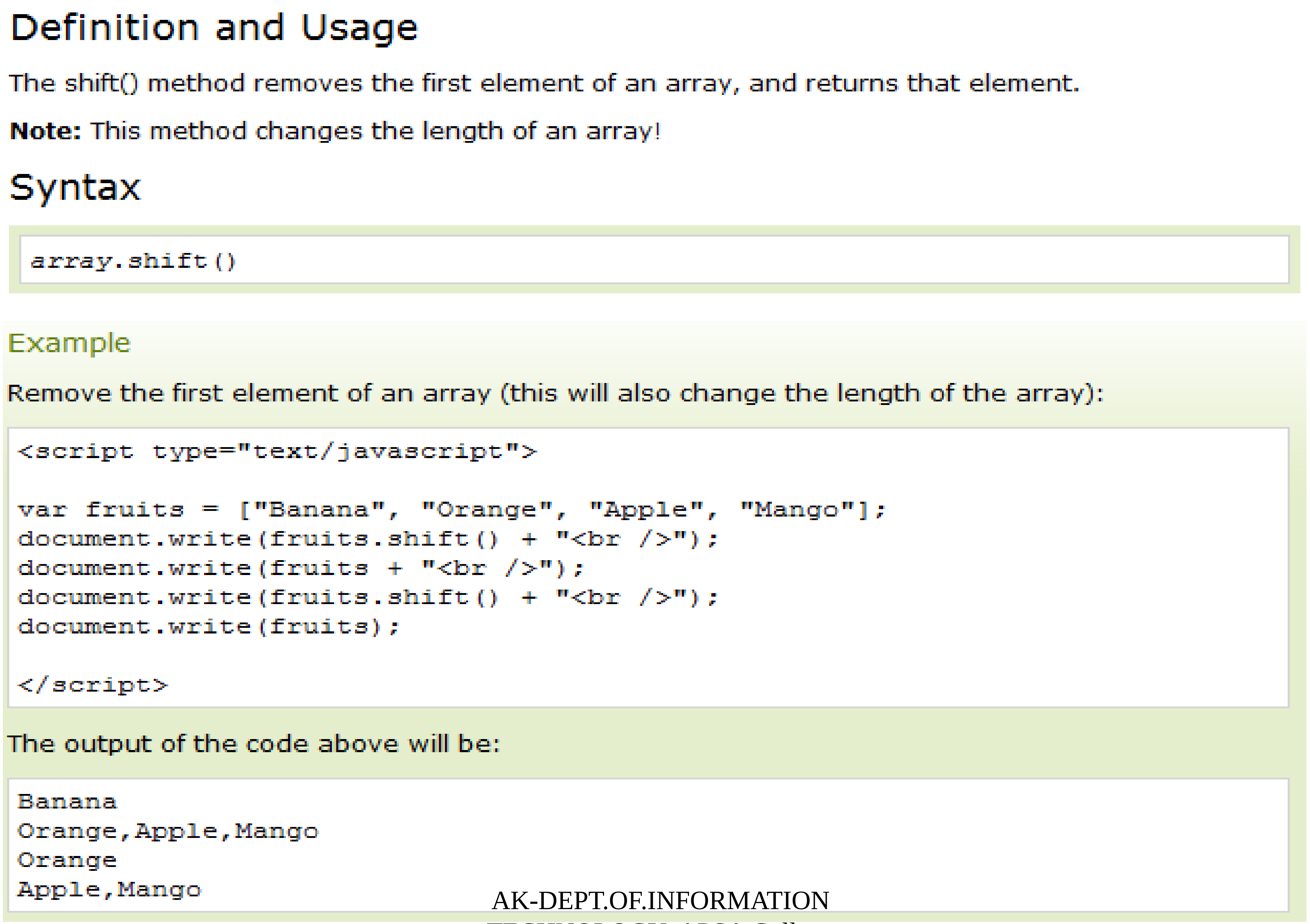
Reverse the order of the elements in an array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.reverse());

</script>
```

The output of the code above will be:

```
Mango,Apple,Orange,Banana
```

JavaScript shift() Method



Definition and Usage

The shift() method removes the first element of an array, and returns that element.

Note: This method changes the length of an array!

Syntax

```
array.shift()
```

Example

Remove the first element of an array (this will also change the length of the array):

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.shift() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.shift() + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Banana
Orange,Apple,Mango
Orange
Apple,Mango
```

JavaScript slice() Method

Definition and Usage

The slice() method selects a part of an array, and returns the new array.

Note: The original array will not be changed.

Syntax

```
array.slice(start, end)
```

Parameter	**Description**
start	Required. An integer that specifies where to start the selection (The first element has an index of 0). You can also use negative numbers to select from the end of an array
end	Optional. An integer that specifies where to end the selection. If omitted, slice() selects all elements from the start position and to the end of the array

Example

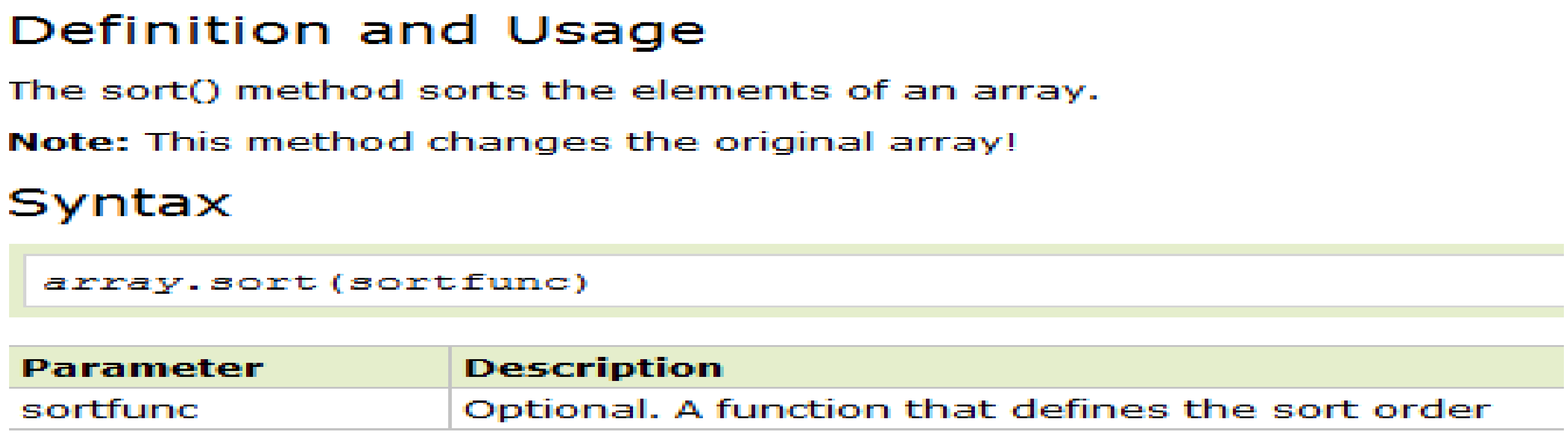
Select elements from an array, and return the new arrays:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.slice(0,1) + "<br />");
document.write(fruits.slice(1) + "<br />");
document.write(fruits.slice(-2) + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Banana
Orange,Apple,Mango
Apple,Mango
Banana,Orange,Apple,Mango
```

JavaScript sort() Method



Definition and Usage

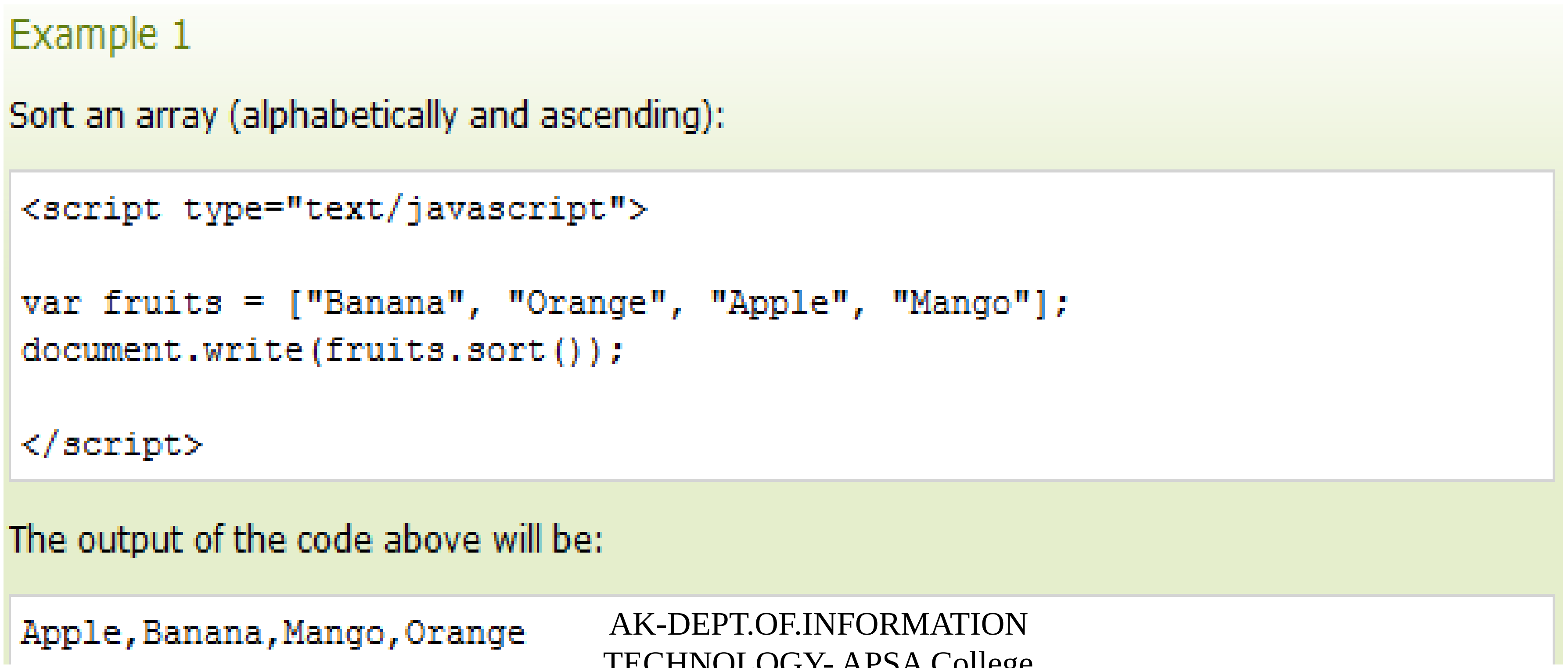
The sort() method sorts the elements of an array.

Note: This method changes the original array!

Syntax

```
array.sort(sortfunc)
```

Parameter	Description
sortfunc	Optional. A function that defines the sort order



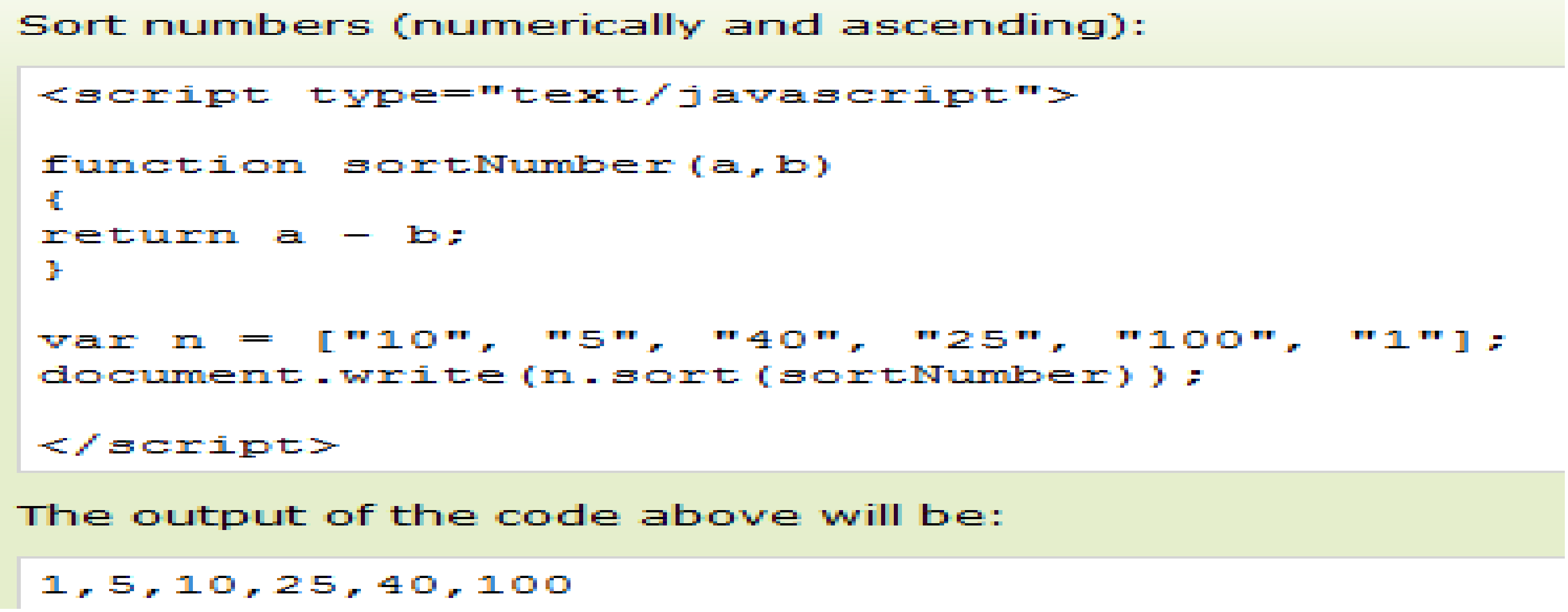
Example 1

Sort an array (alphabetically and ascending):

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());

</script>
```

The output of the code above will be:

```
Apple,Banana,Mango,Orange
```

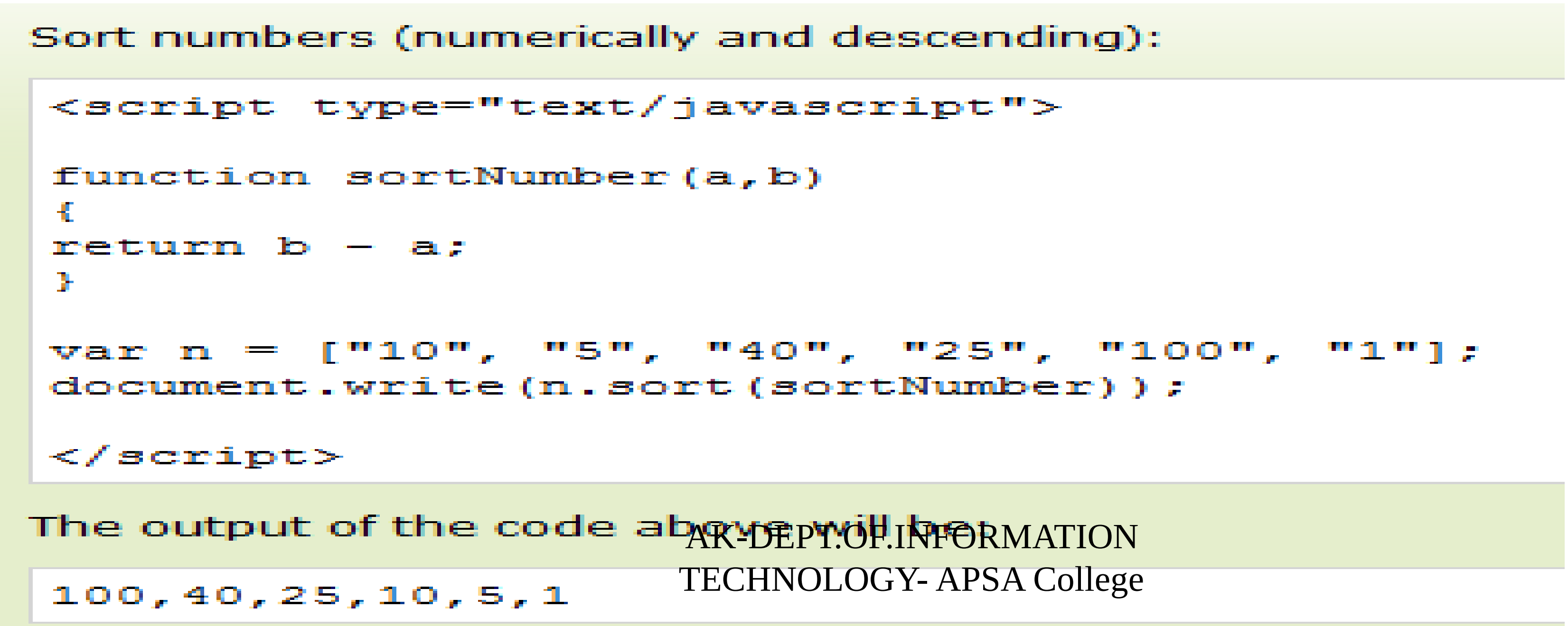


Sort numbers (numerically and ascending):

```
<script type="text/javascript">

function sortNumber(a,b)
{
return a - b;
}

var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));

</script>
```

The output of the code above will be:

```
1,5,10,25,40,100
```



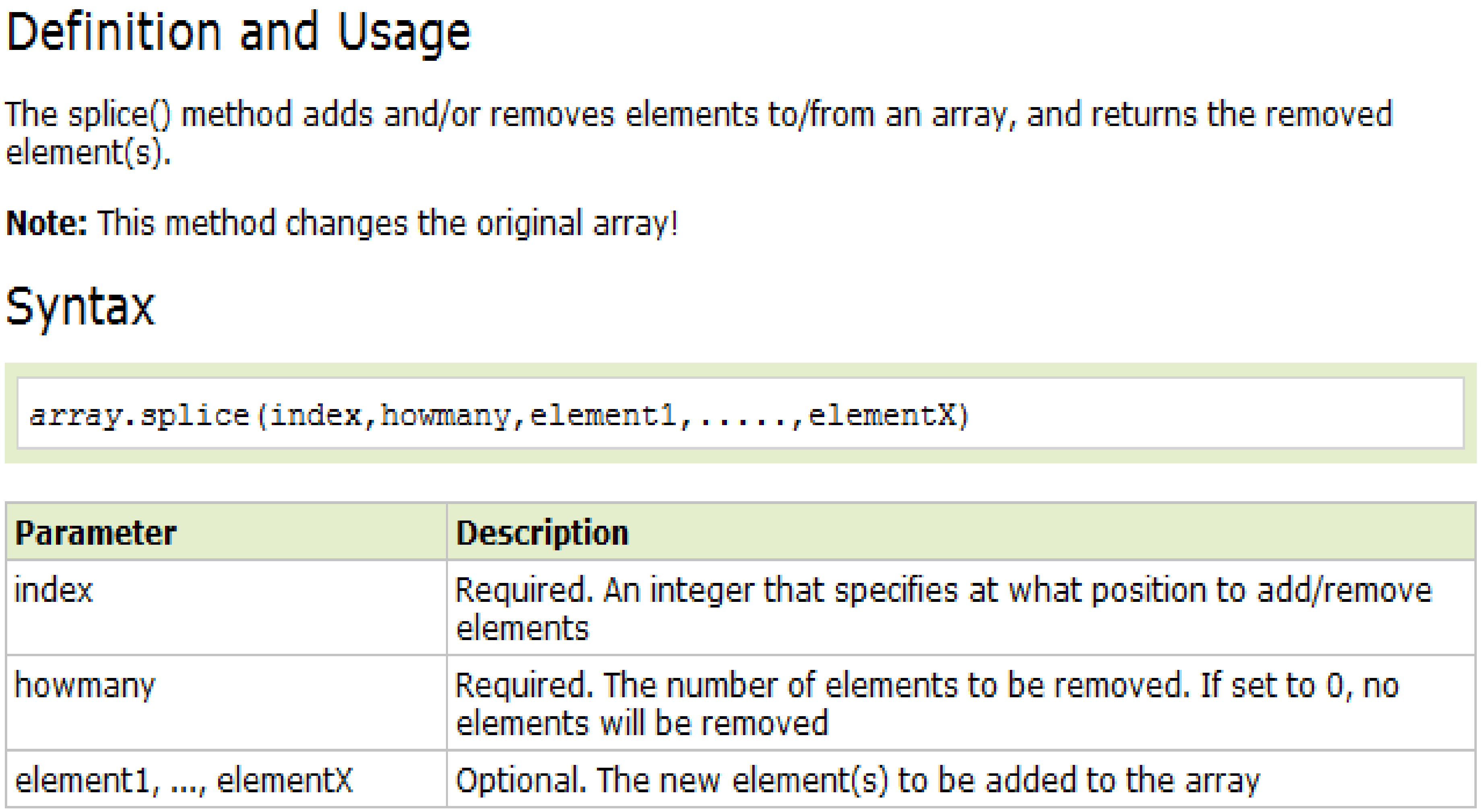
Sort numbers (numerically and descending):

```
<script type="text/javascript">

function sortNumber(a,b)
{
return b - a;
}

var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));

</script>
```

The output of the code above will be:

```
100,40,25,10,5,1
```

JavaScript splice() Method



Definition and Usage

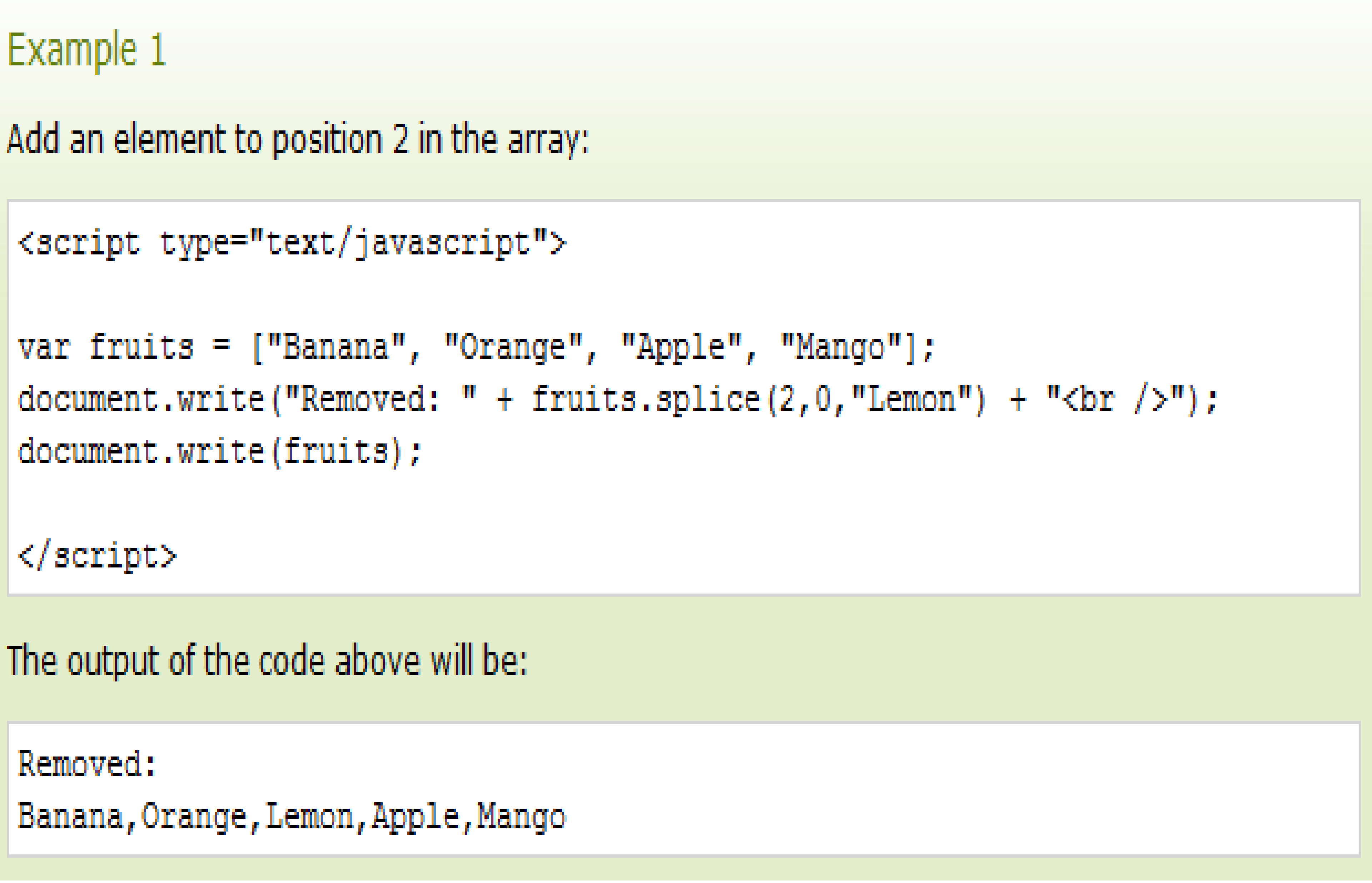
The splice() method adds and/or removes elements to/from an array, and returns the removed element(s).

Note: This method changes the original array!

Syntax

```
array.splice(index,howmany,element1,.....,elementX)
```

Parameter	Description
index	Required. An integer that specifies at what position to add/remove elements
howmany	Required. The number of elements to be removed. If set to 0, no elements will be removed
element1, ..., elementX	Optional. The new element(s) to be added to the array



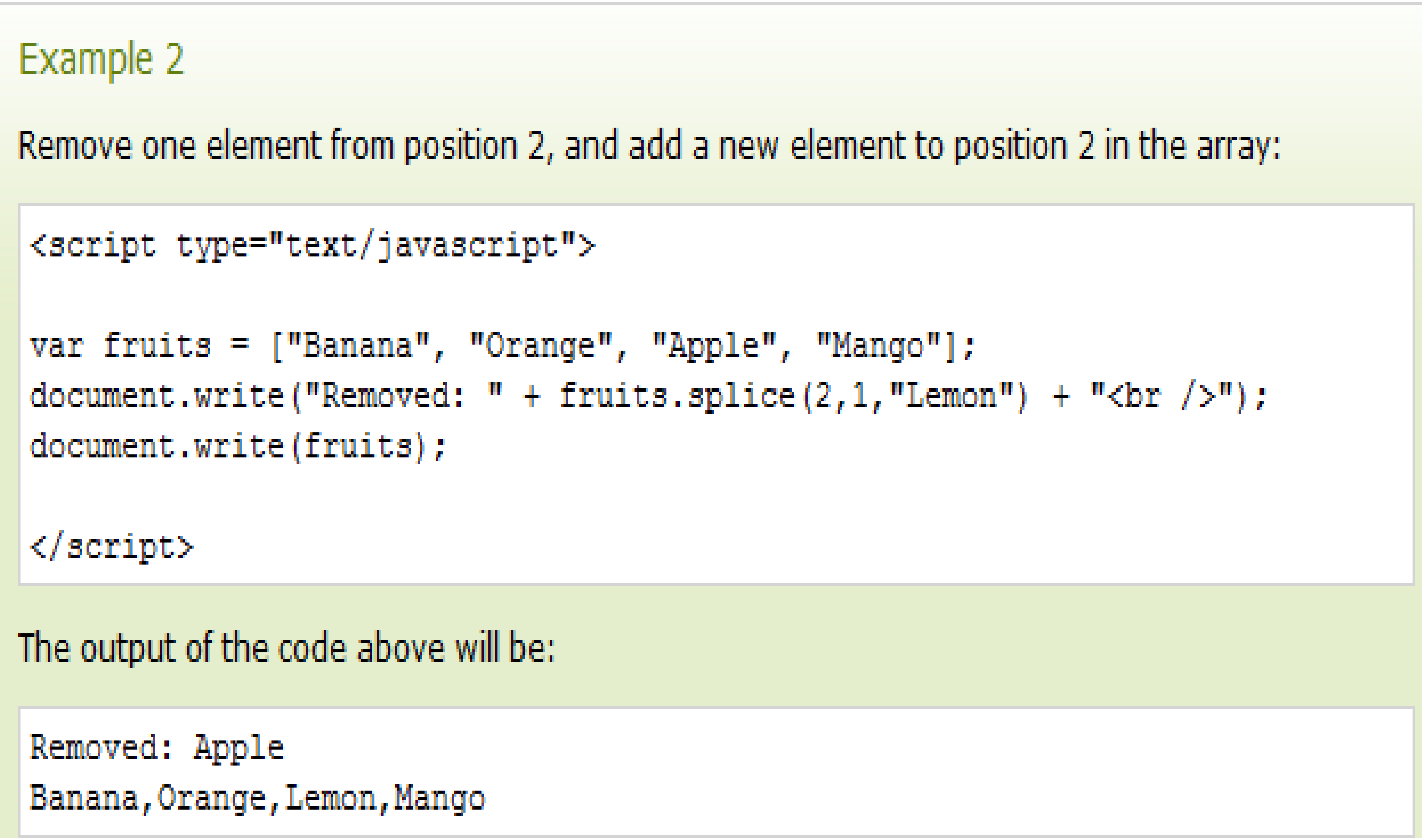
Example 1

Add an element to position 2 in the array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Removed: " + fruits.splice(2,0,"Lemon") + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Removed:
Banana,Orange,Lemon,Apple,Mango
```



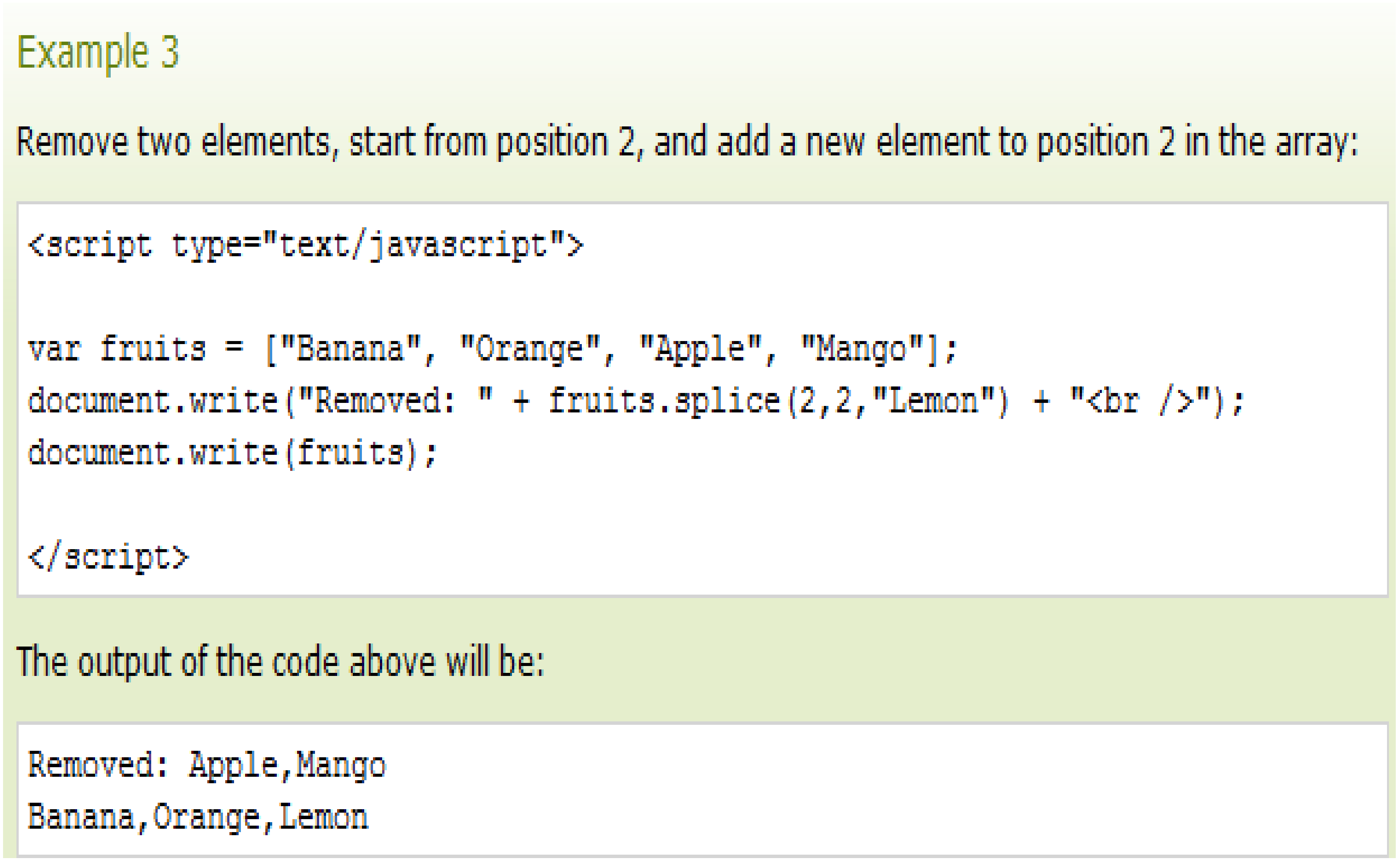
Example 2

Remove one element from position 2, and add a new element to position 2 in the array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Removed: " + fruits.splice(2,1,"Lemon") + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Removed: Apple
Banana,Orange,Lemon,Mango
```



Example 3

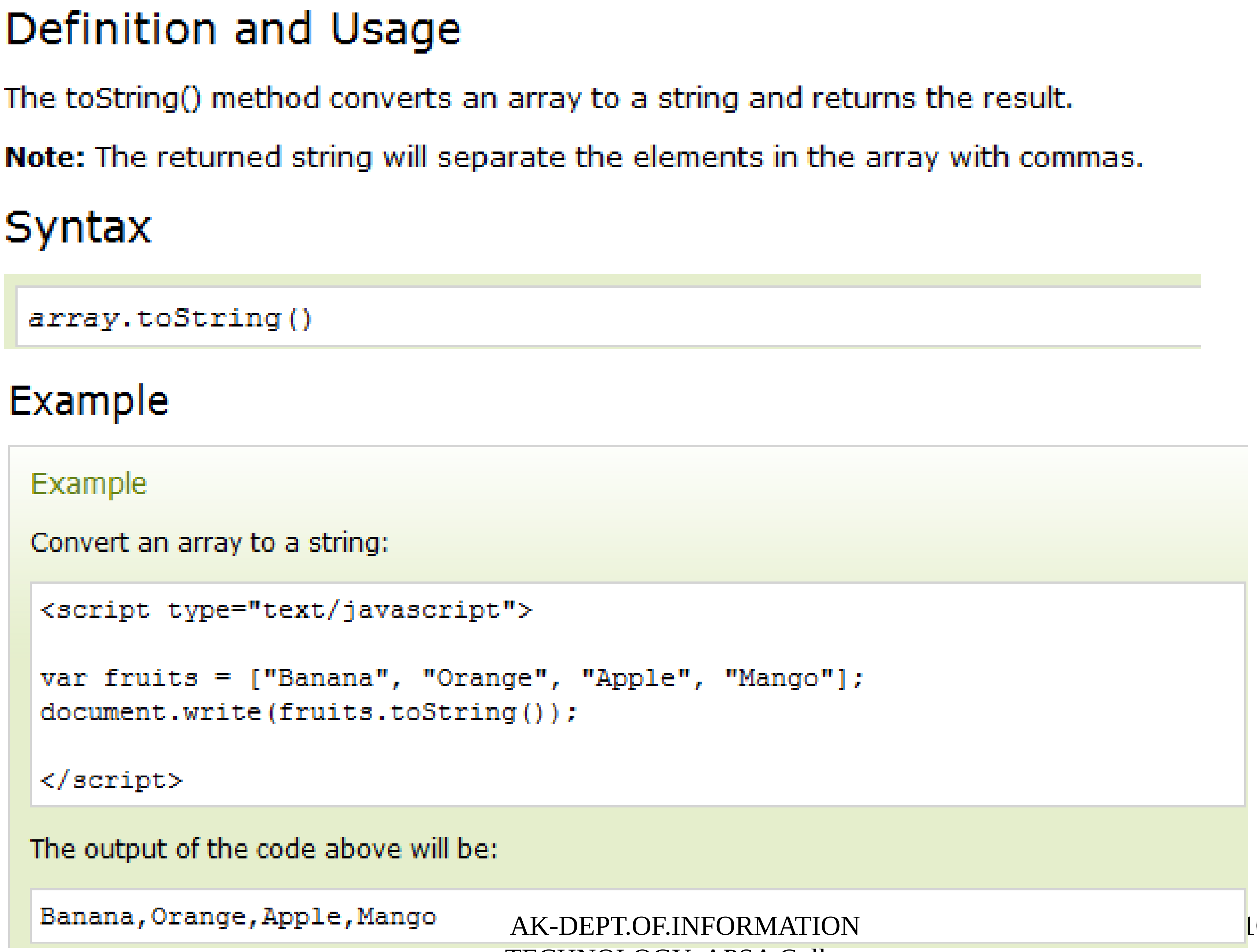
Remove two elements, start from position 2, and add a new element to position 2 in the array:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Removed: " + fruits.splice(2,2,"Lemon") + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
Removed: Apple,Mango
Banana,Orange,Lemon
```

JavaScript toString() Method



Definition and Usage

The toString() method converts an array to a string and returns the result.

Note: The returned string will separate the elements in the array with commas.

Syntax

```
array.toString()
```

Example

Example

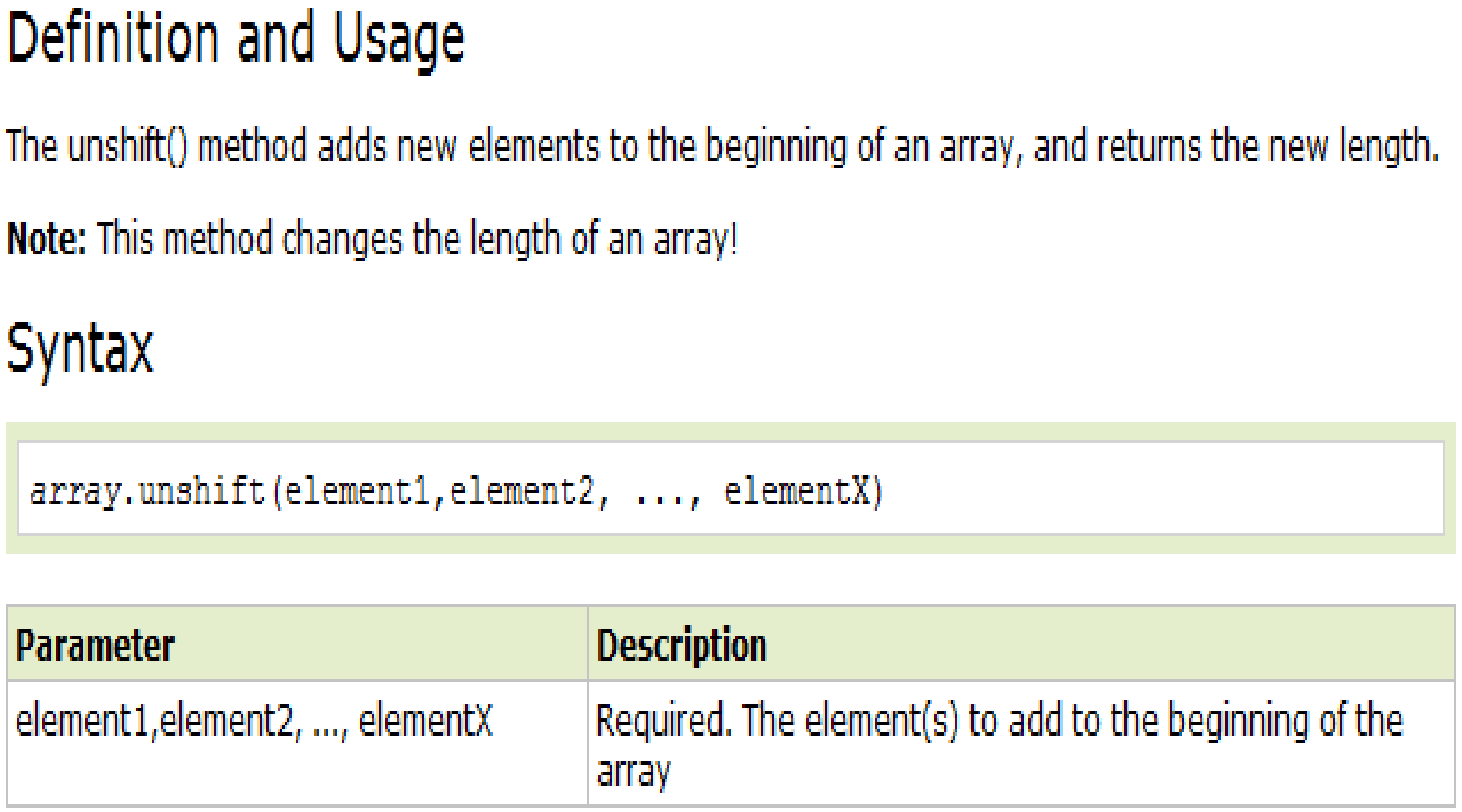
Convert an array to a string:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.toString());

</script>
```

The output of the code above will be:

```
Banana,Orange,Apple,Mango
```

JavaScript unshift() Method



Definition and Usage

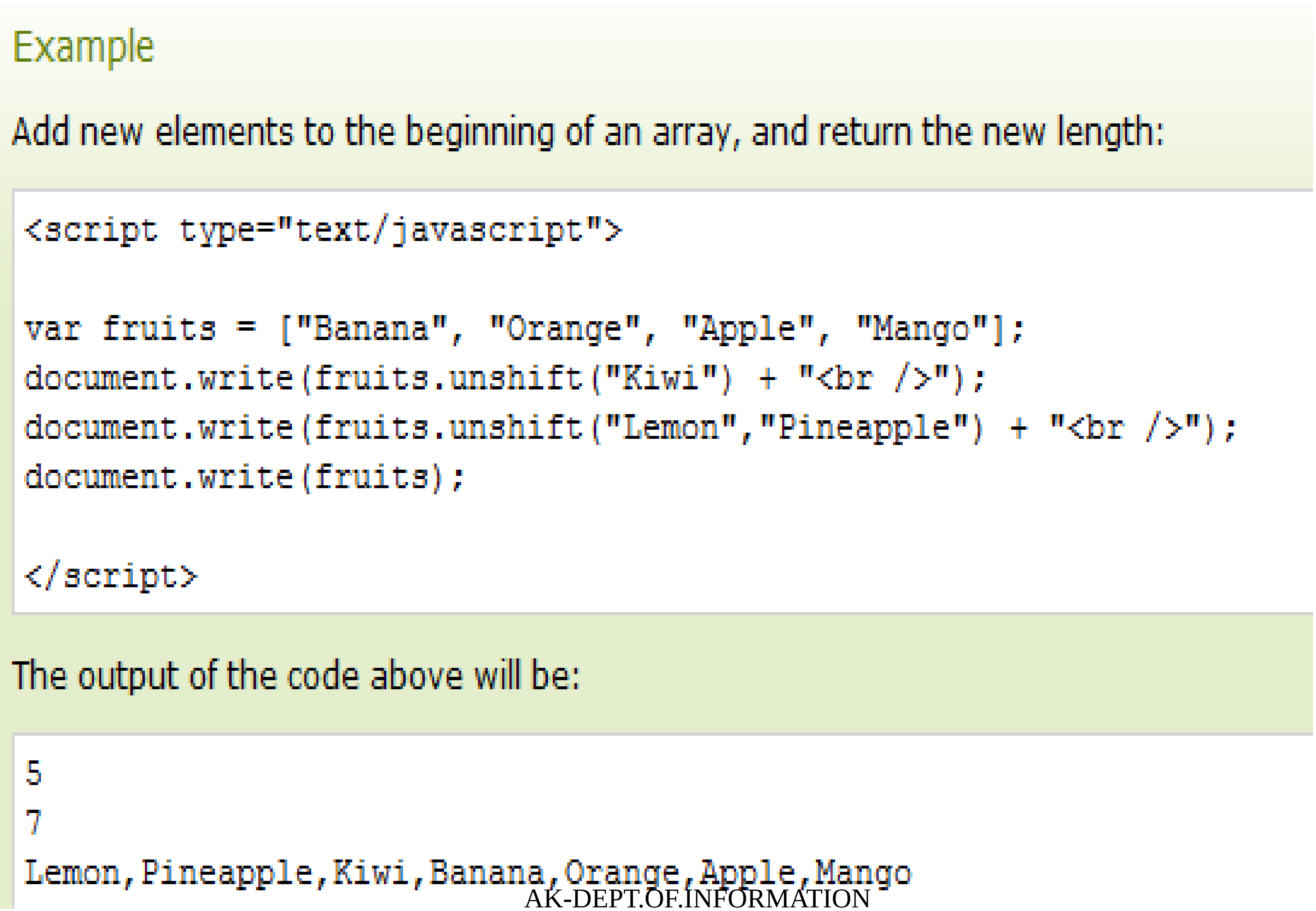
The unshift() method adds new elements to the beginning of an array, and returns the new length.

Note: This method changes the length of an array!

Syntax

```
array.unshift(element1,element2, ..., elementX)
```

Parameter	Description
element1,element2, ..., elementX	Required. The element(s) to add to the beginning of the array



Example

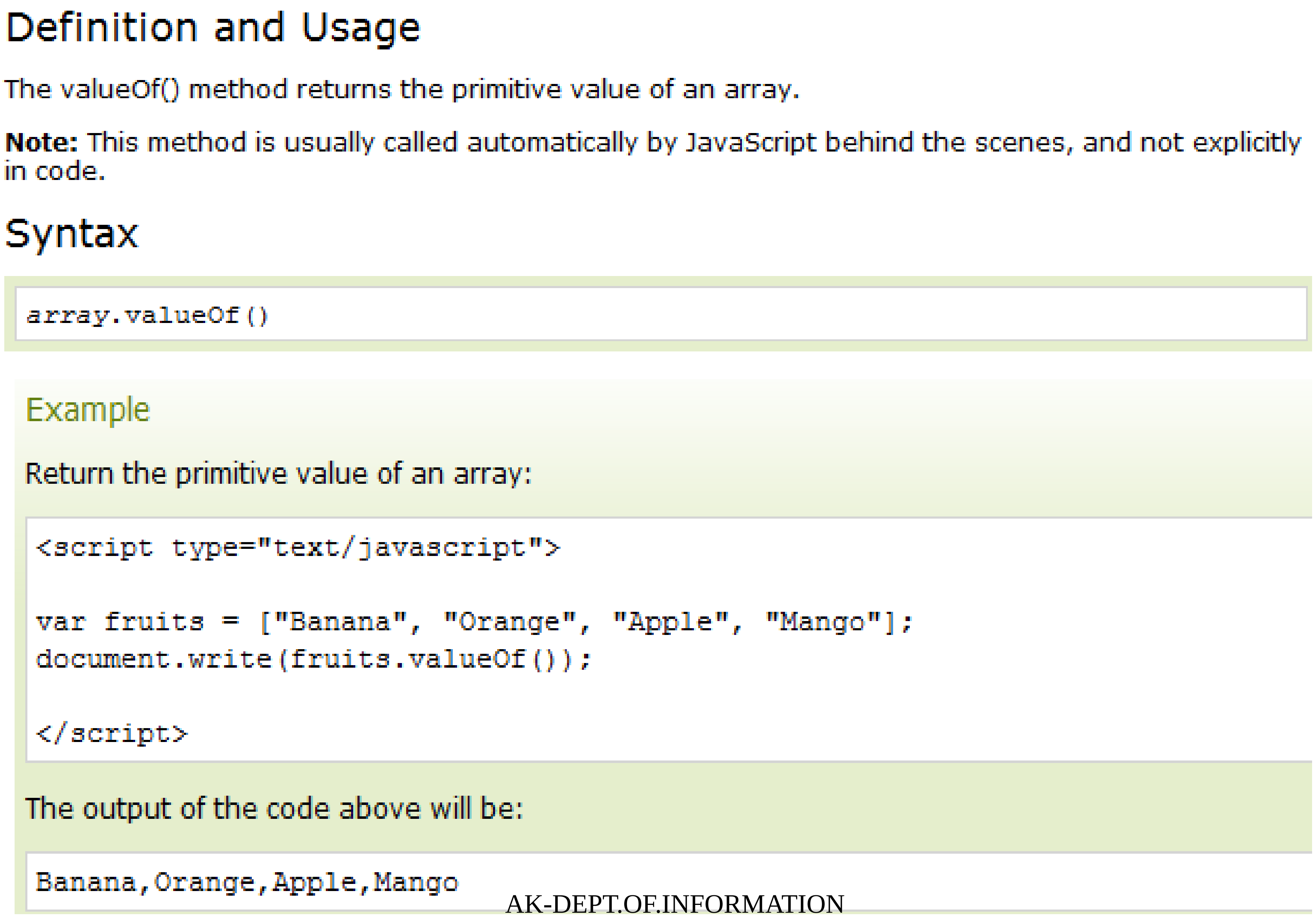
Add new elements to the beginning of an array, and return the new length:

```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.unshift("Kiwi") + "<br />");
document.write(fruits.unshift("Lemon","Pineapple") + "<br />");
document.write(fruits);

</script>
```

The output of the code above will be:

```
5
7
Lemon,Pineapple,Kiwi,Banana,Orange,Apple,Mango
```

JavaScript valueOf() Method



Definition and Usage

The valueOf() method returns the primitive value of an array.

Note: This method is usually called automatically by JavaScript behind the scenes, and not explicitly in code.

Syntax

```
array.valueOf()
```

Example

Return the primitive value of an array:

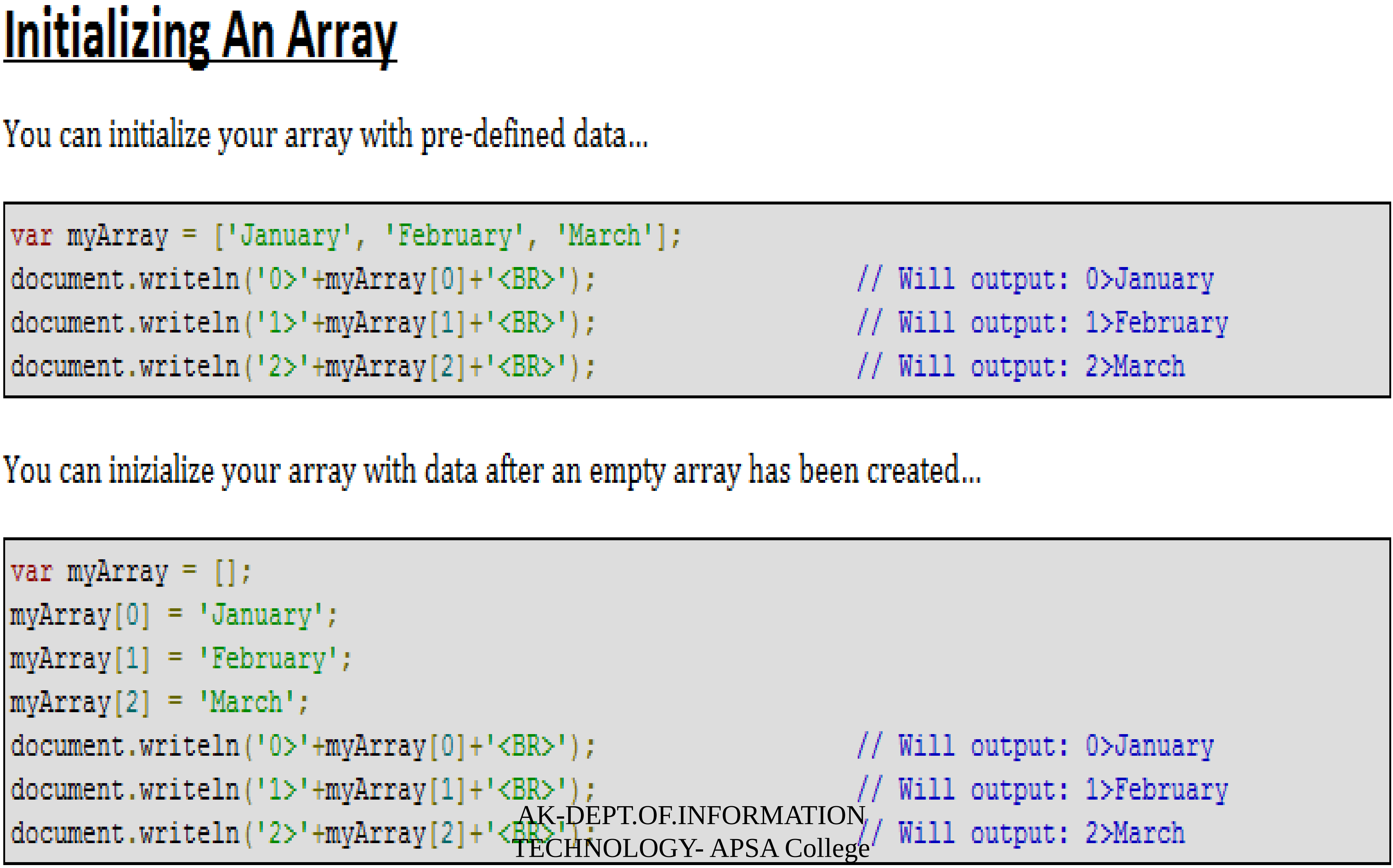
```
<script type="text/javascript">

var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.valueOf());

</script>
```

The output of the code above will be:

```
Banana,Orange,Apple,Mango
```

var badArray = new Array(10); // Creates an empty Array that's sized for 10 elements.

var goodArray= [10];   // Creates an Array with 10 as the first element.



Initializing An Array

You can initialize your array with pre-defined data...

```
var myArray = ['January', 'February', 'March'];
document.writeln('0>'+myArray[0]+'<BR>');                // Will output: 0>January
document.writeln('1>'+myArray[1]+'<BR>');                // Will output: 1>February
document.writeln('2>'+myArray[2]+'<BR>');                // Will output: 2>March
```

You can inizialize your array with data after an empty array has been created...

```
var myArray = [];
myArray[0] = 'January';
myArray[1] = 'February';
myArray[2] = 'March';
document.writeln('0>'+myArray[0]+'<BR>');                // Will output: 0>January
document.writeln('1>'+myArray[1]+'<BR>');                // Will output: 1>February
document.writeln('2>'+myArray[2]+'<BR>');                // Will output: 2>March
```

If you skip an element, the "blank" Array elements will be of type "undefined"

```
var myArray = [];
myArray[0] = 'January';
myArray[1] = 'February';
myArray[5] = 'March';
document.writeln('0>'+myArray[0]+'<BR>');                   // Will output: 0>January
document.writeln('1>'+myArray[1]+'<BR>');                   // Will output: 1>February
document.writeln('2>'+myArray[2]+'<BR>');                   // Will output: 2>undefined
document.writeln('3>'+myArray[3]+'<BR>');                   // Will output: 3>undefined
document.writeln('4>'+myArray[4]+'<BR>');                   // Will output: 4>undefined
document.writeln('5>'+myArray[5]+'<BR>');                   // Will output: 5>March
```

Storing Data In An Array

An array can store anything you can assign to a variable: booleans, numbers, strings, functions, objects, other Arrays, even regular expressions...

```
var myArray = [ 3, 'hello!', function() {return 5}, {'color':'blue', 'budget':25}, /[ell]/i ];
document.writeln('0>'+myArray[0]+'<BR>');                   // Will output: 0>3
document.writeln('1>'+myArray[1]+'<BR>');                   // Will output: 1>hello!
document.writeln('2>'+myArray[2]()+'<BR>');                 // Will output: 2>5
document.writeln('3>'+myArray[3].color+'<BR>');             // Will output: 3>blue
document.writeln('3>'+myArray[3].budget+'<BR>');            // Will output: 3>25
document.writeln('4>'+myArray[4].test(myArray[1])+'<BR>');  // Will output: 4>true
```

Multi-Dimensional Arrays

Since an Array can store other Arrays you can get the benefit of multi-dimension arrays.

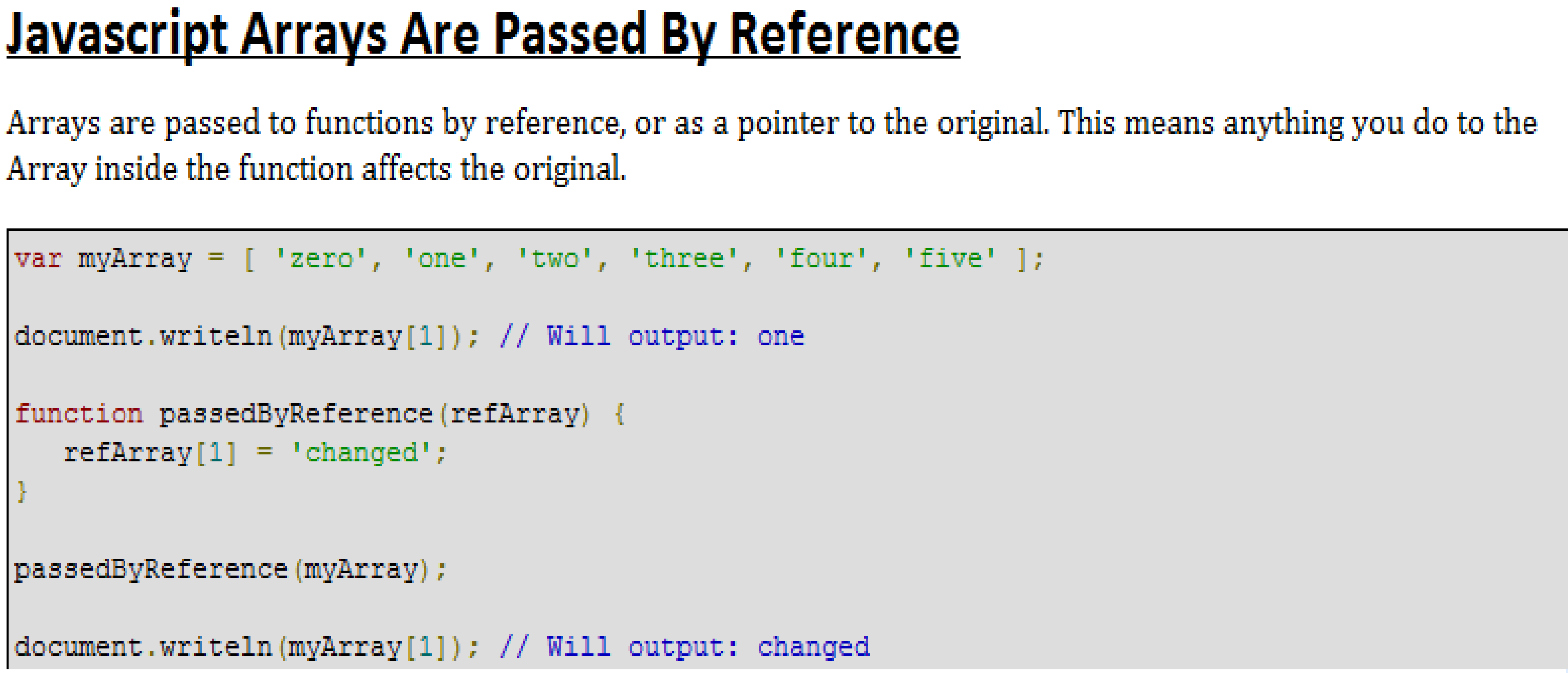
```
var x=[0,1,2,3,4,5];
var y=[x];
```

In the above example we created an array named "x" and assigned it as the first element in the array "y". If we ask for the value of y[0] it will return the contents of "x" as a string because we didn't specify an index.

```
var x=[0,1,2,3,4,5];
var y=[x];
document.writeln(y[0]); // Will output: 0,1,2,3,4,5
```

If we wanted the third index we'd access it this way...

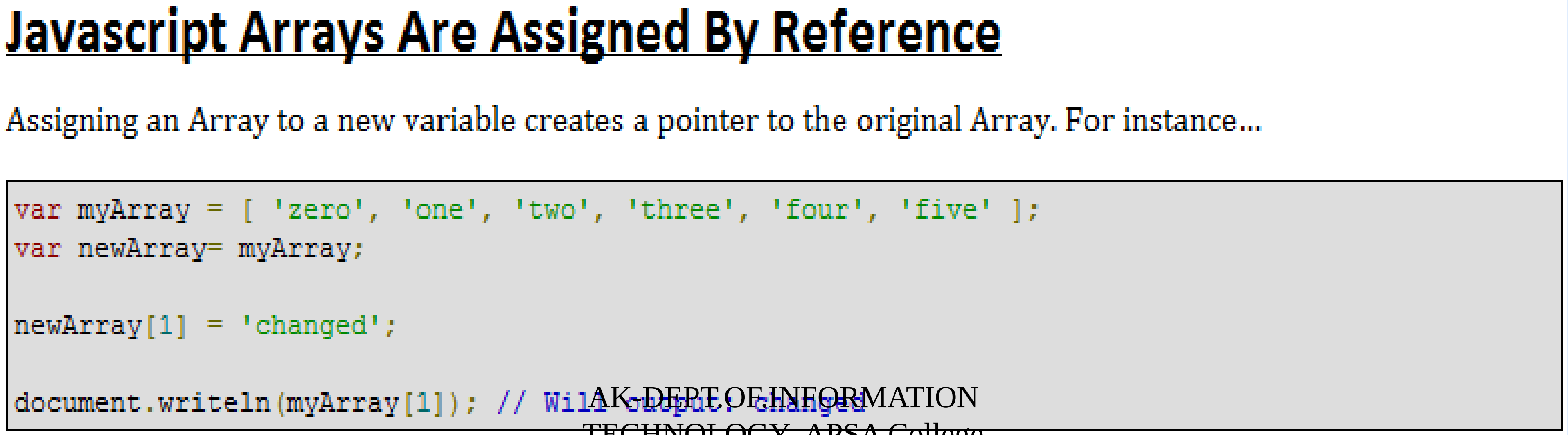
```
var x=[0,1,2,3,4,5];
var y=[x];
document.writeln(y[0][3]); // Will output: 2
```



Javascript Arrays Are Passed By Reference

Arrays are passed to functions by reference, or as a pointer to the original. This means anything you do to the Array inside the function affects the original.

```
var myArray = [ 'zero', 'one', 'two', 'three', 'four', 'five' ];

document.writeln(myArray[1]); // Will output: one

function passedByReference(refArray) {
   refArray[1] = 'changed';
}

passedByReference(myArray);

document.writeln(myArray[1]); // Will output: changed
```



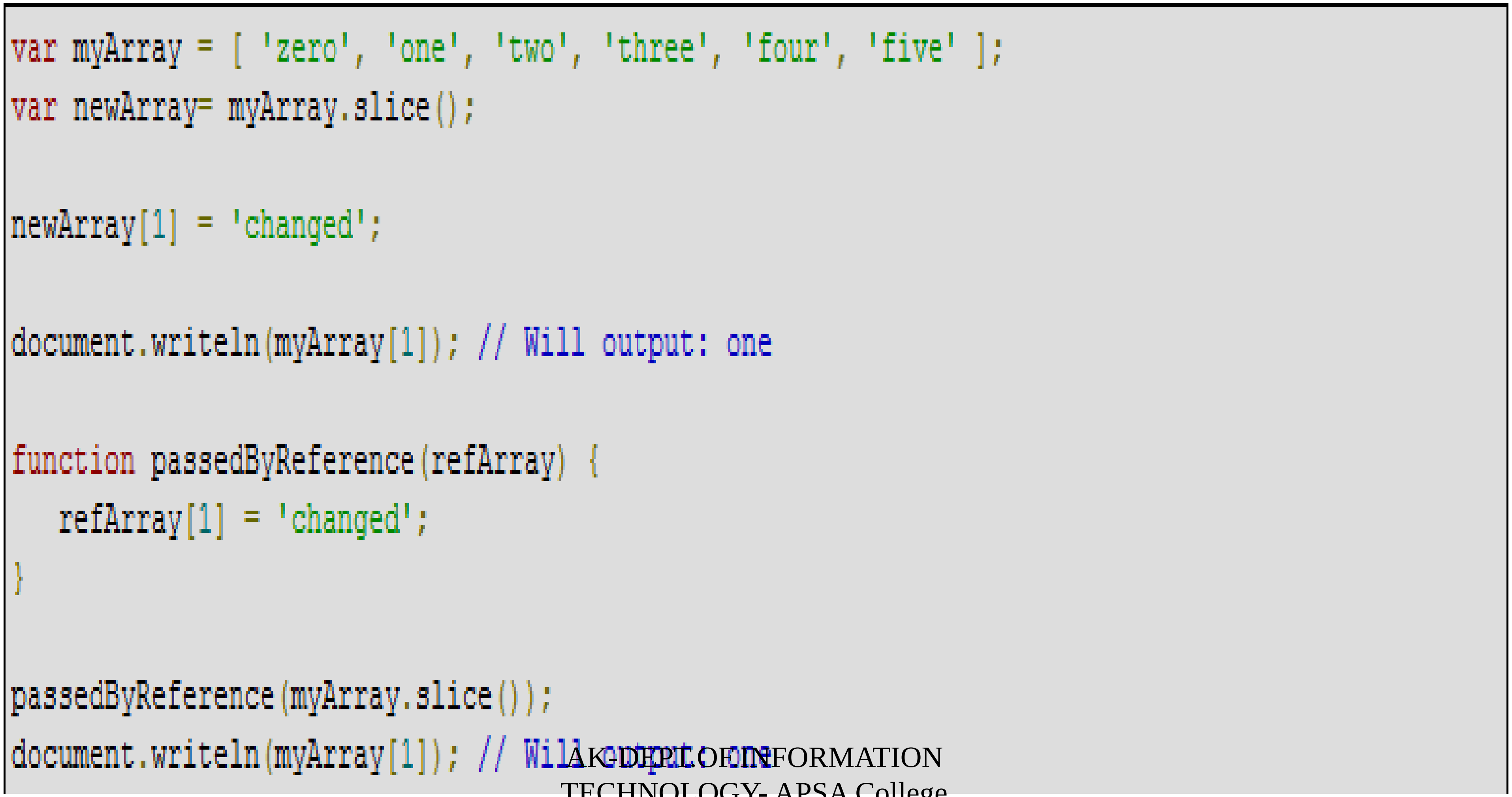
Javascript Arrays Are Assigned By Reference

Assigning an Array to a new variable creates a pointer to the original Array. For instance...

```
var myArray = [ 'zero', 'one', 'two', 'three', 'four', 'five' ];
var newArray= myArray;

newArray[1] = 'changed';

document.writeln(myArray[1]); // Will output: changed
```

Passing Arrays As Values

To pass an Array by *value* instead of by reference, use the Array.slice() method.



```
var myArray = [ 'zero', 'one', 'two', 'three', 'four', 'five' ];
var newArray= myArray.slice();

newArray[1] = 'changed';

document.writeln(myArray[1]); // Will output: one

function passedByReference(refArray) {
   refArray[1] = 'changed';
}

passedByReference(myArray.slice());
document.writeln(myArray[1]); // Will output: one
```

Javascript Does Not Support Associative Arrays

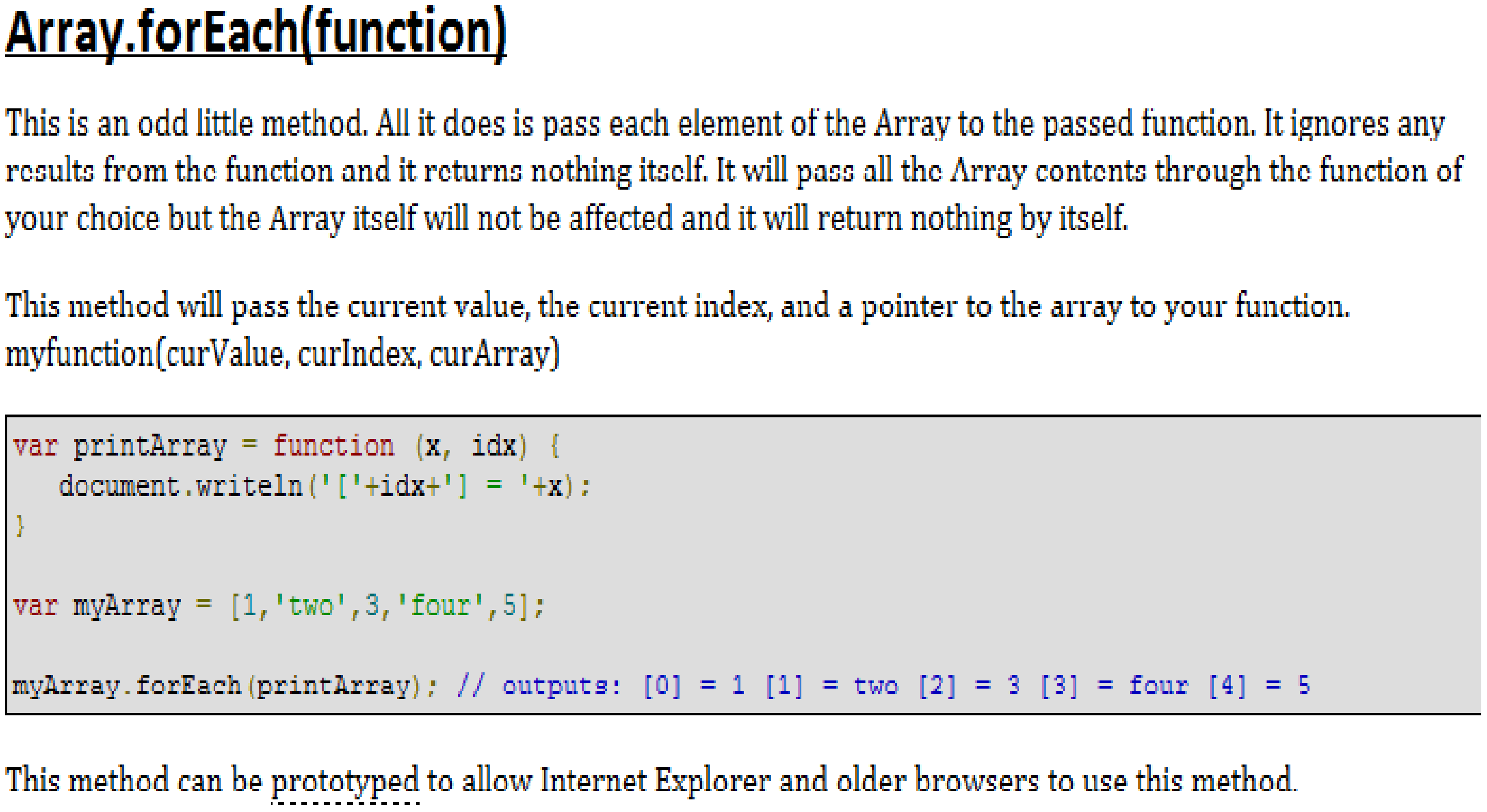
An associative array is an array which uses a string instead of a number as an index.

```
var normalArray    = [];
    normalArray[1] = 'This is an enumerated array';

    alert(normalArray[1]);   // outputs: This is an enumerated array

var associativeArray           = [];
    associativeArray['person'] = 'John Smith';

    alert(associativeArray['person']); // outputs: John Smith
```

Javascript does not have, and does not support Associative Arrays. However… All arrays in Javascript are objects and Javascript's object syntax gives a basic emulation of an associative Array. For this reason the example code above will actually work. Be warned that this is not a real array and it has real pitfalls if you try to use it. The 'person' element in the example becomes part of the Array object's properties and methods, just like .length, .sort(), .splice(), and all the other built-in properties and methods.

You can loop through an object's properties with the following loop…

```
var associativeArray = [];
associativeArray["one"] = "First";
associativeArray["two"] = "Second";
associativeArray["three"] = "Third";
for (i in associativeArray) {
   document.writeln(i+':'+associativeArray[i]+', ');
   // outputs: one:First, two:Second, three:Third
};
```



Array.forEach(function)

This is an odd little method. All it does is pass each element of the Array to the passed function. It ignores any results from the function and it returns nothing itself. It will pass all the Array contents through the function of your choice but the Array itself will not be affected and it will return nothing by itself.

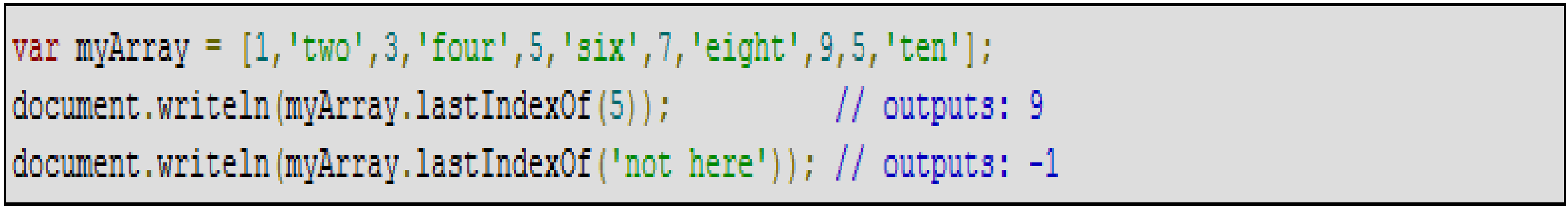
This method will pass the current value, the current index, and a pointer to the array to your function.
myfunction(curValue, curIndex, curArray)

```
var printArray = function (x, idx) {
   document.writeln('['+idx+'] = '+x);
}

var myArray = [1,'two',3,'four',5];

myArray.forEach(printArray); // outputs: [0] = 1 [1] = two [2] = 3 [3] = four [4] = 5
```

This method can be prototyped to allow Internet Explorer and older browsers to use this method.

Array.lastIndexOf(searchStr[, startIndex])

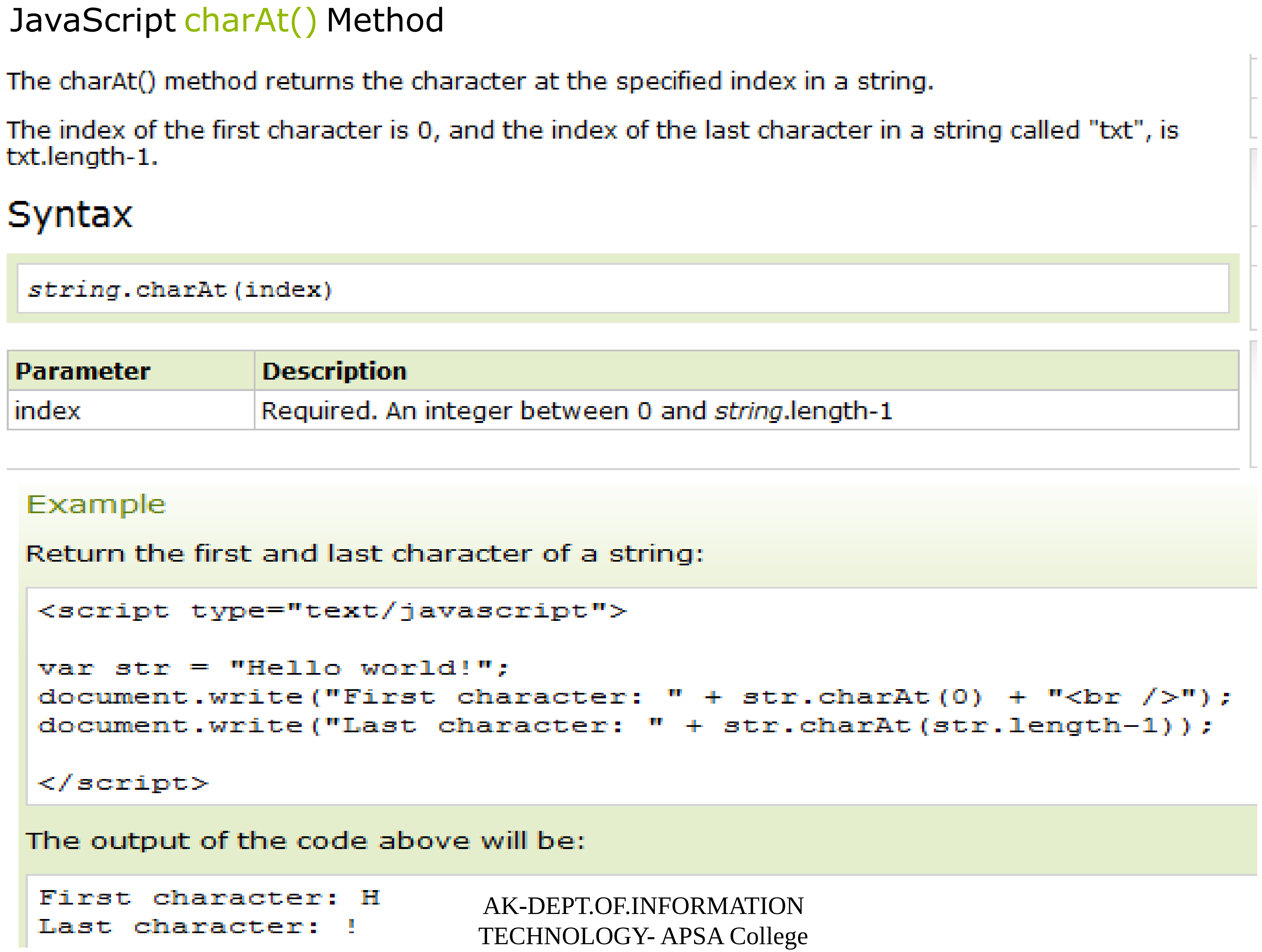
Array.indexOf() searches from first to last, lastIndexOf searches from last to first.



```
var myArray = [1,'two',3,'four',5,'six',7,'eight',9,5,'ten'];
document.writeln(myArray.lastIndexOf(5));          // outputs: 9
document.writeln(myArray.lastIndexOf('not here')); // outputs: -1
```

String Object Methods

Method	Description
charAt()	Returns the character at the specified index
charCodeAt()	Returns the Unicode of the character at the specified index
concat()	Joins two or more strings, and returns a copy of the joined strings
fromCharCode()	Converts Unicode values to characters
indexOf()	Returns the position of the first found occurrence of a specified value in a string
lastIndexOf()	Returns the position of the last found occurrence of a specified value in a string
match()	Searches for a match between a regular expression and a string, and returns the matches
replace()	Searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring
search()	Searches for a match between a regular expression and a string, and returns the position of the match
slice()	Extracts a part of a string and returns a new string
split()	Splits a string into an array of substrings
substr()	Extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	Extracts the characters from a string, between two specified indices
toLowerCase()	Converts a string to lowercase letters
toUpperCase()	Converts a string to uppercase letters
valueOf()	Returns the primitive value of a String object



JavaScript charAt() Method

The charAt() method returns the character at the specified index in a string.

The index of the first character is 0, and the index of the last character in a string called "txt", is txt.length-1.

Syntax

```
string.charAt(index)
```

Parameter	Description
index	Required. An integer between 0 and *string*.length-1

Example

Return the first and last character of a string:

```
<script type="text/javascript">

var str = "Hello world!";
document.write("First character: " + str.charAt(0) + "<br />");
document.write("Last character: " + str.charAt(str.length-1));

</script>
```

The output of the code above will be:

```
First character: H
Last character: !
```

charCodeAt() Method

The charCodeAt() method returns the Unicode of the character at the specified index in a string.

The index of the first character is 0, and the index of the last character in a string called "txt", is txt.length-1.

Syntax

```
string.charCodeAt(index)
```

Parameter	**Description**
index	Required. An integer between 0 and *string*.length-1

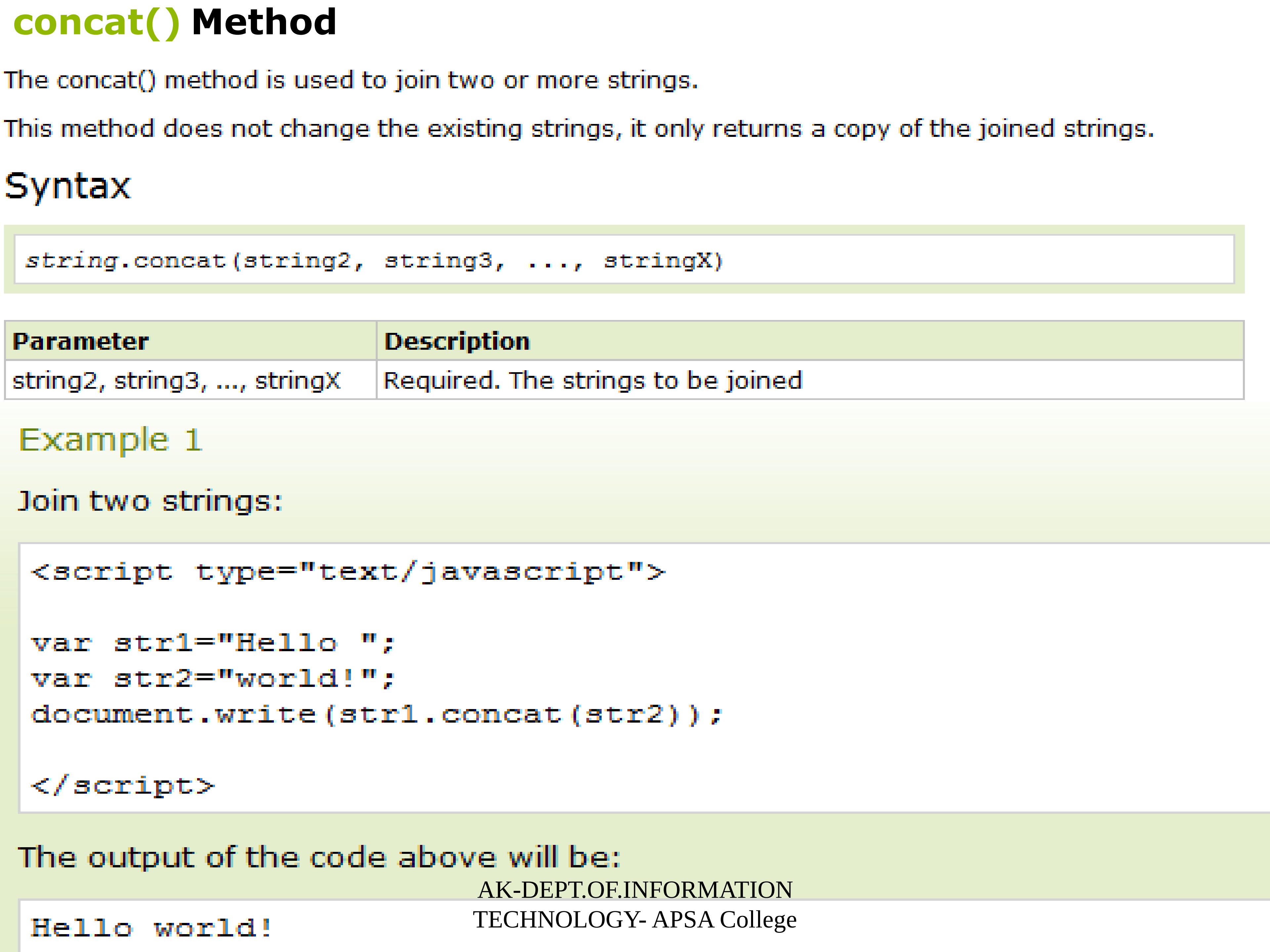
Example

Return the Unicode of the first and last character in a string:

```
<script type="text/javascript">

var str = "Hello world!";
document.write("First character: " + str.charCodeAt(0) + "<br />");
document.write("Last character: " + str.charCodeAt(str.length-1));

</script>
```

The output of the code above will be:

```
First character: 72
Last character: 33
```



concat() Method

The concat() method is used to join two or more strings.

This method does not change the existing strings, it only returns a copy of the joined strings.

Syntax

```
string.concat(string2, string3, ..., stringX)
```

Parameter	Description
string2, string3, ..., stringX	Required. The strings to be joined

Example 1

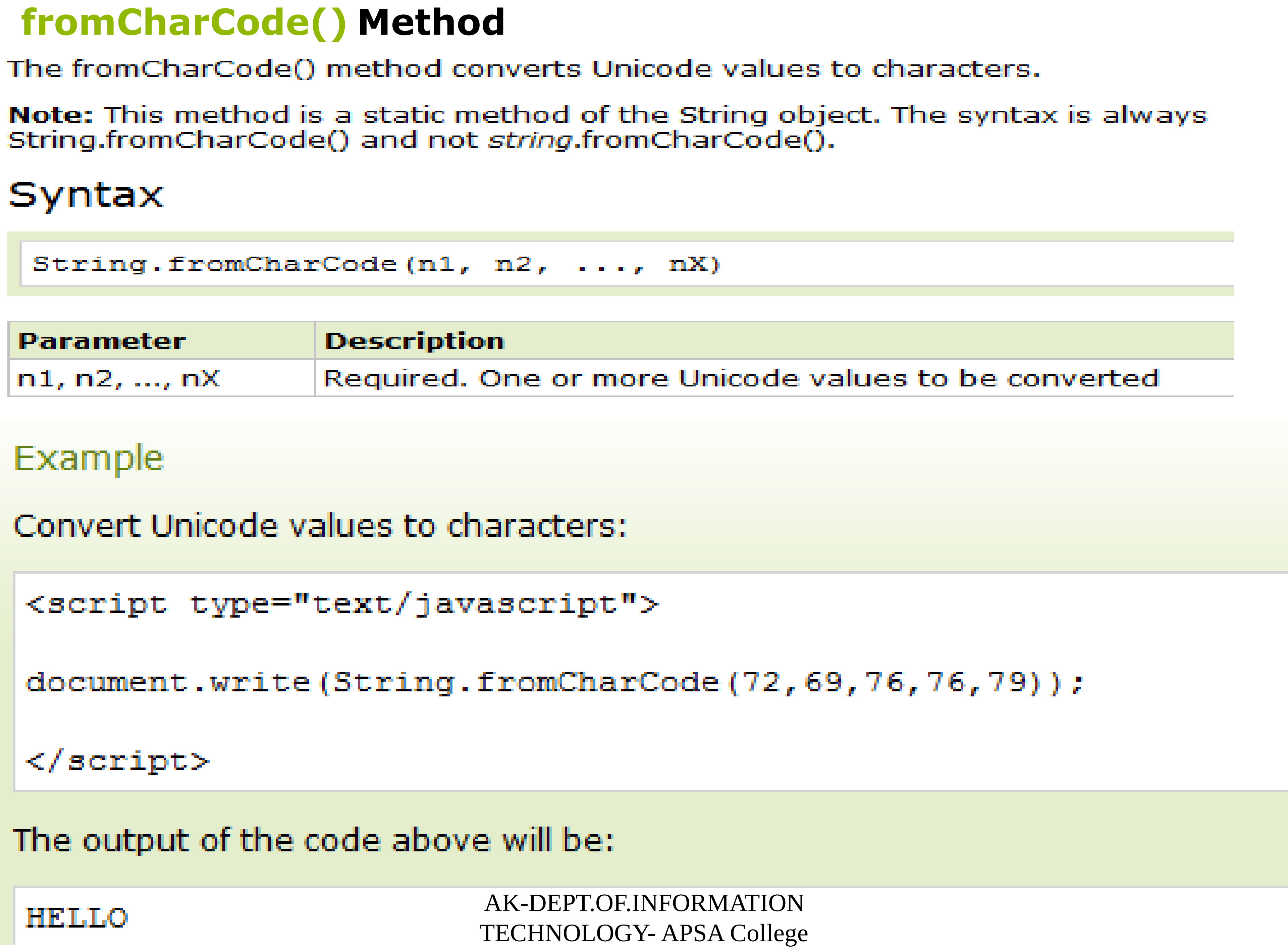
Join two strings:

```
<script type="text/javascript">

var str1="Hello ";
var str2="world!";
document.write(str1.concat(str2));

</script>
```

The output of the code above will be:

```
Hello world!
```

fromCharCode() Method



The fromCharCode() method converts Unicode values to characters.

Note: This method is a static method of the String object. The syntax is always String.fromCharCode() and not *string*.fromCharCode().

Syntax

```
String.fromCharCode(n1, n2, ..., nX)
```

Parameter	Description
n1, n2, ..., nX	Required. One or more Unicode values to be converted

Example

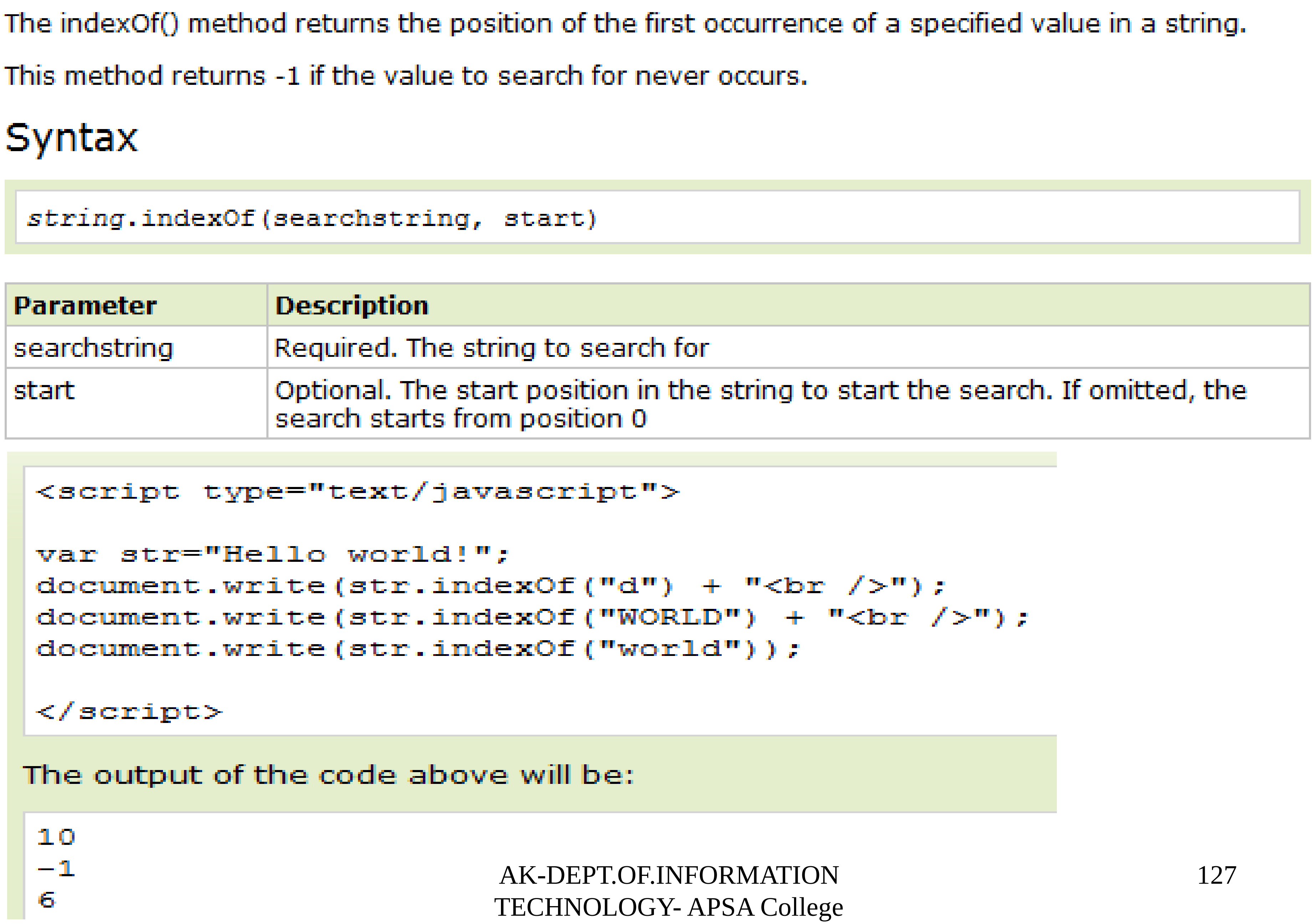
Convert Unicode values to characters:

```
<script type="text/javascript">

document.write(String.fromCharCode(72,69,76,76,79));

</script>
```

The output of the code above will be:

```
HELLO
```

indexOf() Method



The indexOf() method returns the position of the first occurrence of a specified value in a string.

This method returns -1 if the value to search for never occurs.

Syntax

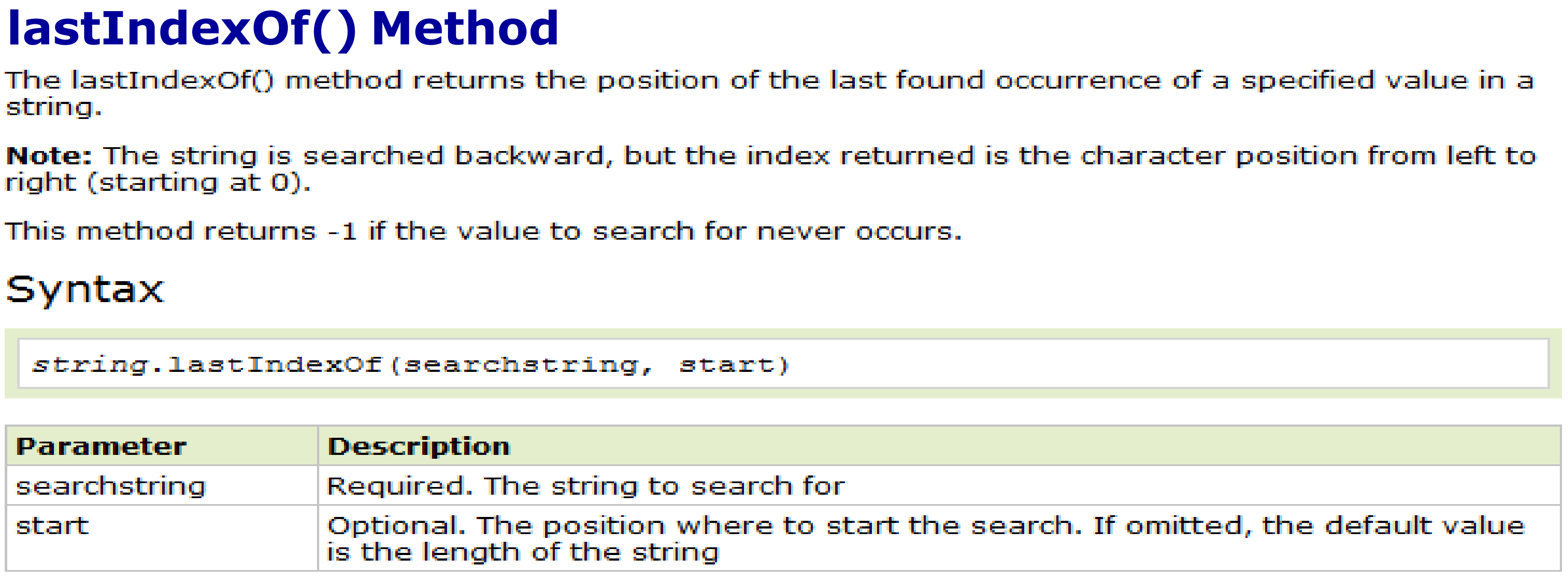
```
string.indexOf(searchstring, start)
```

Parameter	Description
searchstring	Required. The string to search for
start	Optional. The start position in the string to start the search. If omitted, the search starts from position 0

```
<script type="text/javascript">

var str="Hello world!";
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("WORLD") + "<br />");
document.write(str.indexOf("world"));

</script>
```

The output of the code above will be:

```
10
-1
6
```



lastIndexOf() Method

The lastIndexOf() method returns the position of the last found occurrence of a specified value in a string.

Note: The string is searched backward, but the index returned is the character position from left to right (starting at 0).

This method returns -1 if the value to search for never occurs.

Syntax

```
string.lastIndexOf(searchstring, start)
```

Parameter	Description
searchstring	Required. The string to search for
start	Optional. The position where to start the search. If omitted, the default value is the length of the string

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!";
document.write(str.lastIndexOf("o") + "<br />");
document.write(str.indexOf("o") + "<br />");
document.write(str.lastIndexOf("world"));
</script>
</body>
</html>
```

Output:

```
7
4
6
```

match() Method

The match() method searches for a match between a regular expression and a string, and returns the matches.

This method returns an array of matches, or null if no match is found.

```
string.match(regexp)
```

Parameter	Description
regexp	Required. A regular expression.

Example

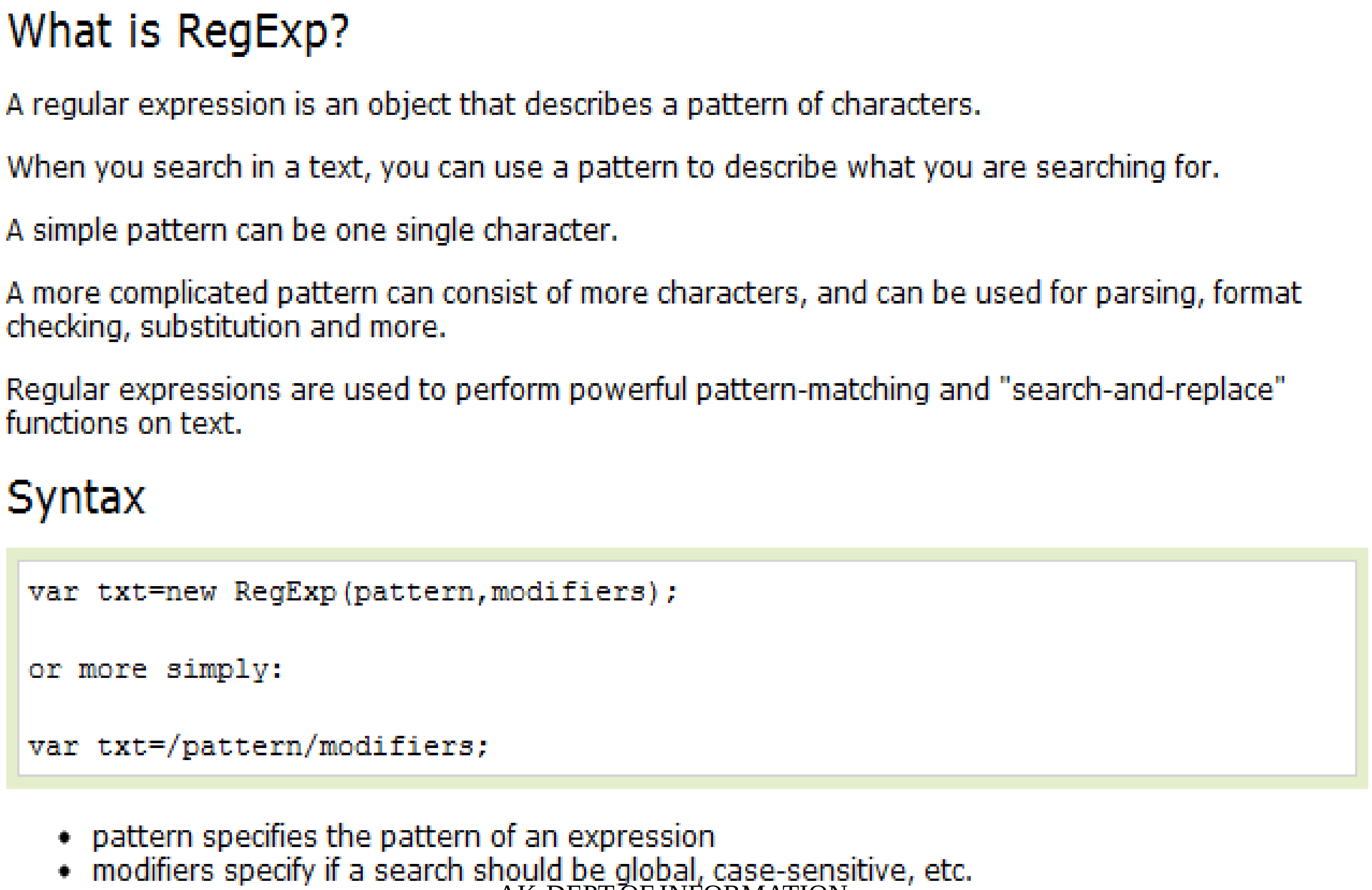
Perform a global, case-insensitive search for "ain":

```
<script type="text/javascript">

var str="The rain in SPAIN stays mainly in the plain";
var patt1=/ain/gi;
document.write(str.match(patt1));

</script>
```

The output of the code above will be:

```
ain,AIN,ain,ain
```

RegExp Object



What is RegExp?

A regular expression is an object that describes a pattern of characters.

When you search in a text, you can use a pattern to describe what you are searching for.

A simple pattern can be one single character.

A more complicated pattern can consist of more characters, and can be used for parsing, format checking, substitution and more.

Regular expressions are used to perform powerful pattern-matching and "search-and-replace" functions on text.

Syntax

```
var txt=new RegExp(pattern,modifiers);

or more simply:

var txt=/pattern/modifiers;
```

- pattern specifies the pattern of an expression
- modifiers specify if a search should be global, case-sensitive, etc.

Example 1 :

```
<html>
<body>
<script type="text/javascript">
var str="Is this all there is?";
var patt1=/is/g;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Output :

```
is,is
```

Example 2 :

```
<html>
<body>
<script type="text/javascript">
var str="Is this all there is?";
var patt1=/is/gi;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Output :

```
Is,is,is
```

test()

The test() method searches a string for a specified value, and returns true or false, depending on the result.

The following example searches a string for the character "e":

Example

```
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
true
```

exec()

The exec() method searches a string for a specified value, and returns the text of the found value. If no match is found, it returns *null*.

The following example searches a string for the character "e":

Example 1

```
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
```

Since there is an "e" in the string, the output of the code above will be:

```
e
```

Modifiers

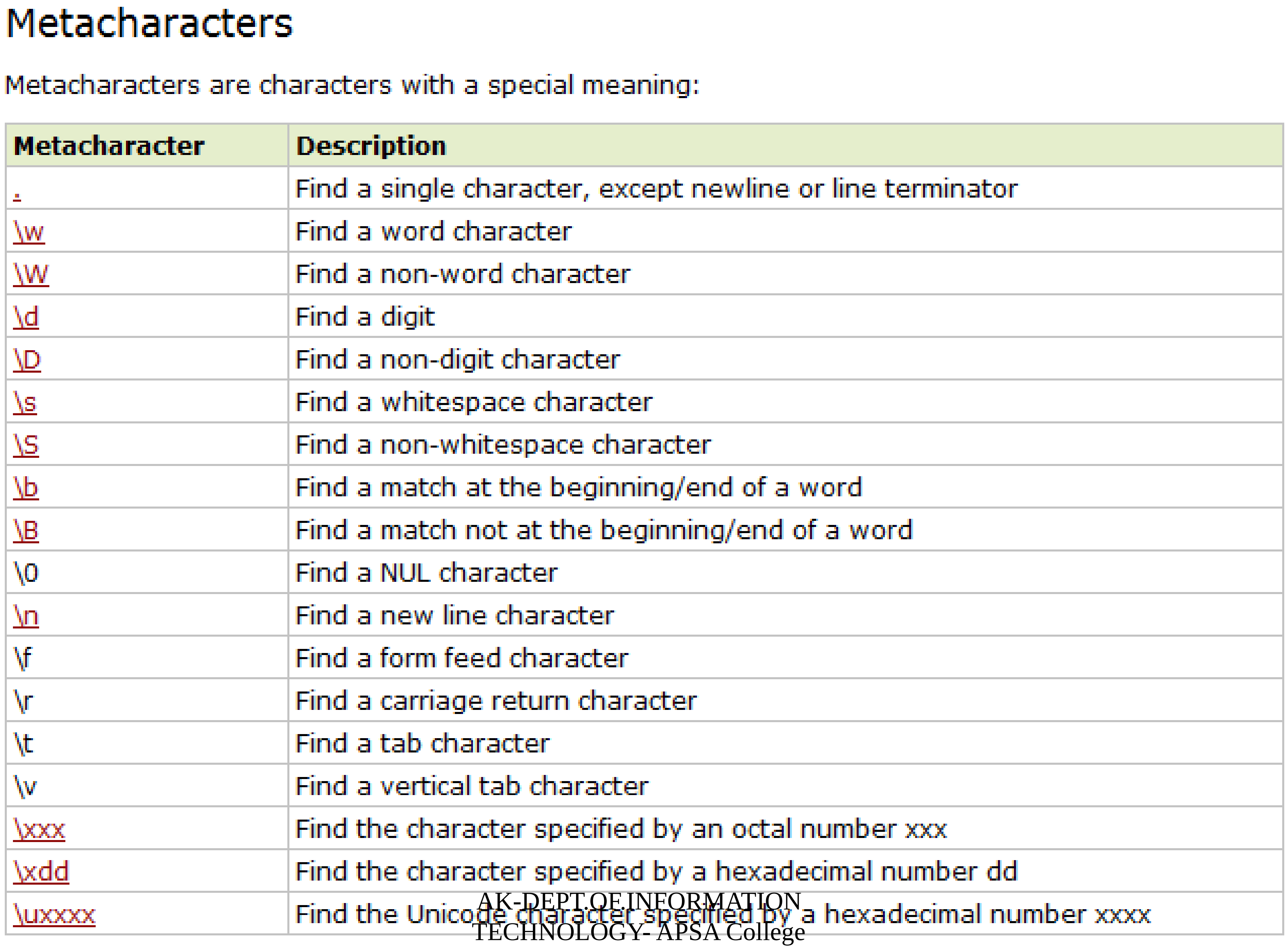
Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[A-Z]	Find any character from uppercase A to uppercase Z
[a-z]	Find any character from lowercase a to lowercase z
[A-z]	Find any character from uppercase A to lowercase z
[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
(red|blue|green)	Find any of the alternatives specified



Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

```
<html>
<body>
<script type="text/javascript">
var str="That's hot!";
var patt1=/h.t/g;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Output

```
hat,hot
```

Quantifiers

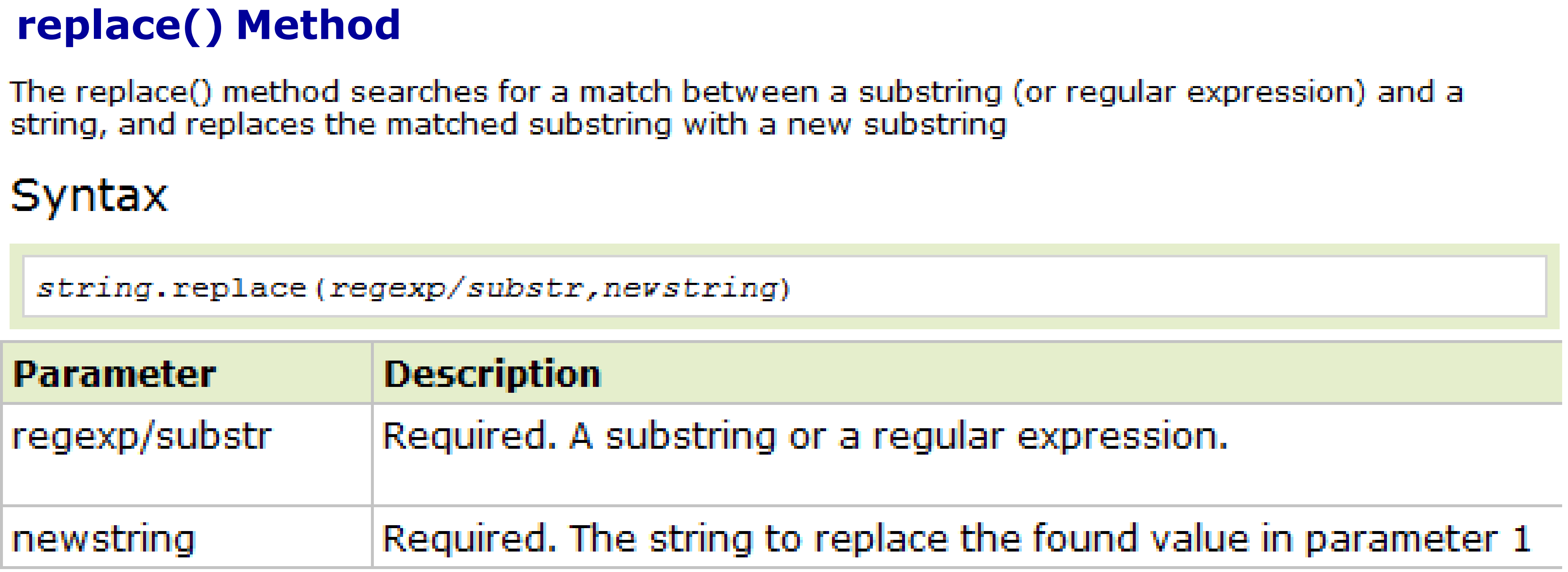
Quantifier	Description
n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X or Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
?=n	Matches any string that is followed by a specific string n
?!n	Matches any string that is not followed by a specific string n

RegExp Object Properties

Property	Description
global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set

RegExp Object Methods

Method	Description
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false



replace() Method

The replace() method searches for a match between a substring (or regular expression) and a string, and replaces the matched substring with a new substring

Syntax

```
string.replace(regexp/substr,newstring)
```

Parameter	Description
regexp/substr	Required. A substring or a regular expression.
newstring	Required. The string to replace the found value in parameter 1

```
<html>
<body>
<script type="text/javascript">
var str="Visit Microsoft!";
document.write(str.replace("Microsoft","Apsa College"));
</script>
</body>
</html>
```

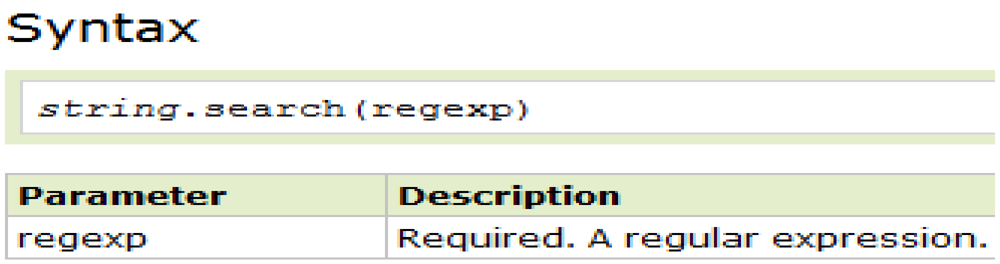
Output :

```
Visit Apsa College!
```

search() Method

The search() method searches for a match between a regular expression and a string.

This method returns the position of the match, or -1 if no match is found.



Syntax

```
string.search(regexp)
```

Parameter	Description
regexp	Required. A regular expression.

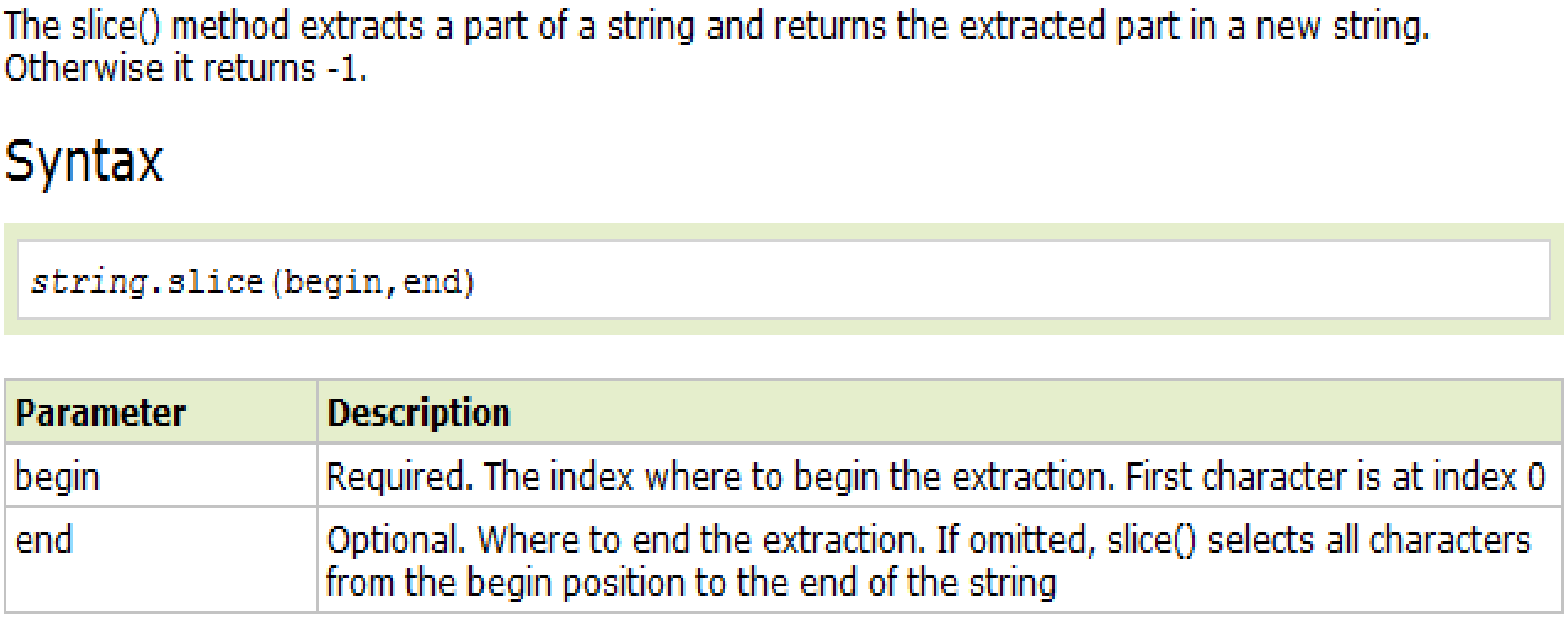
```
<html>
<body>
<script type="text/javascript">
var str="Visit Apsac!";
document.write(str.search("Apsac"));
</script>
</body>
</html>
```

Output : 6

```
<html>
<body>
<script type="text/javascript">
var str="Visit Apsa College!";
document.write(str.search(/college/i));
</script>
</body>
</html>
```

Output : 11

slice() method

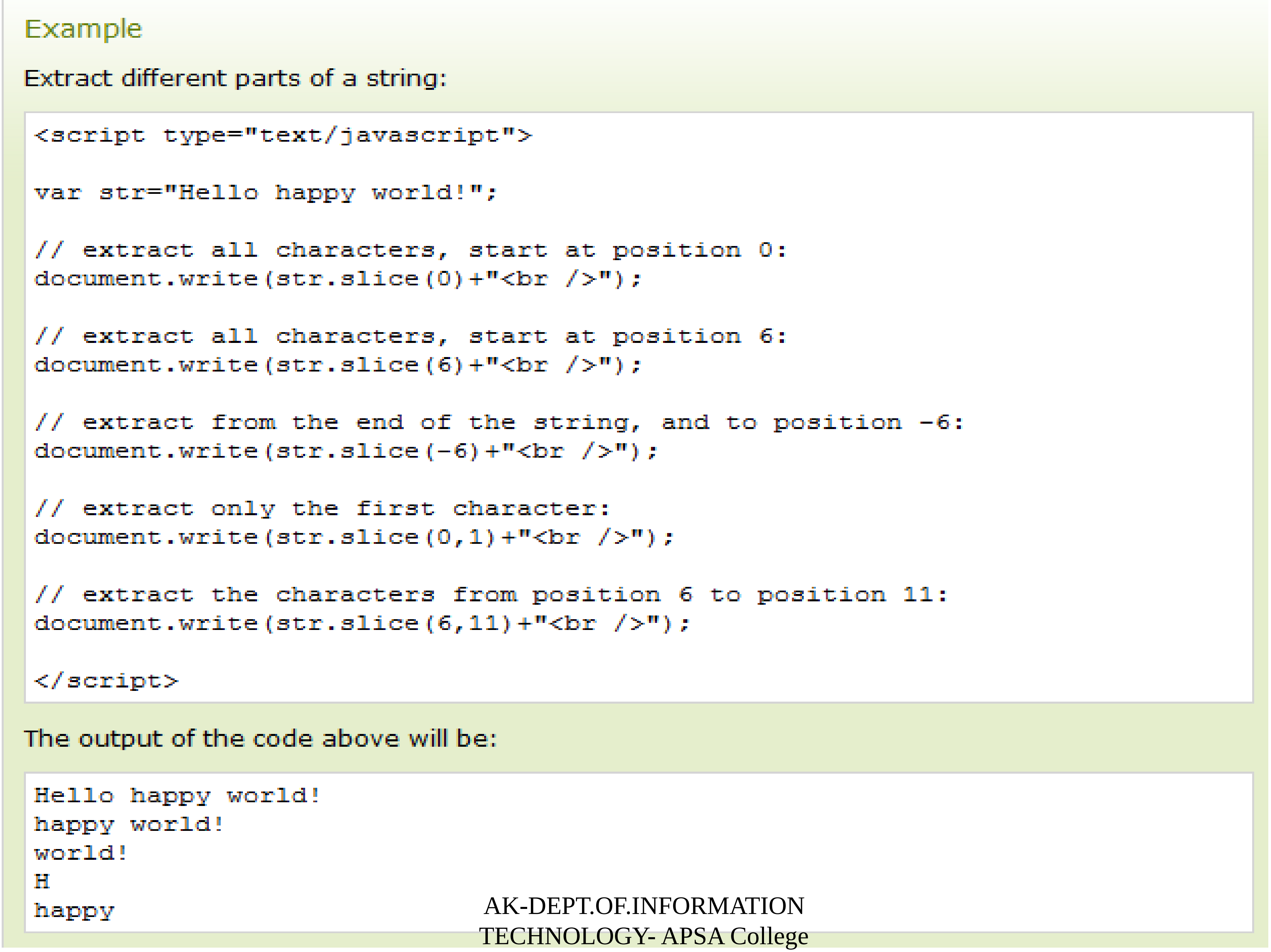


The slice() method extracts a part of a string and returns the extracted part in a new string. Otherwise it returns -1.

Syntax

```
string.slice(begin,end)
```

Parameter	Description
begin	Required. The index where to begin the extraction. First character is at index 0
end	Optional. Where to end the extraction. If omitted, slice() selects all characters from the begin position to the end of the string



Example

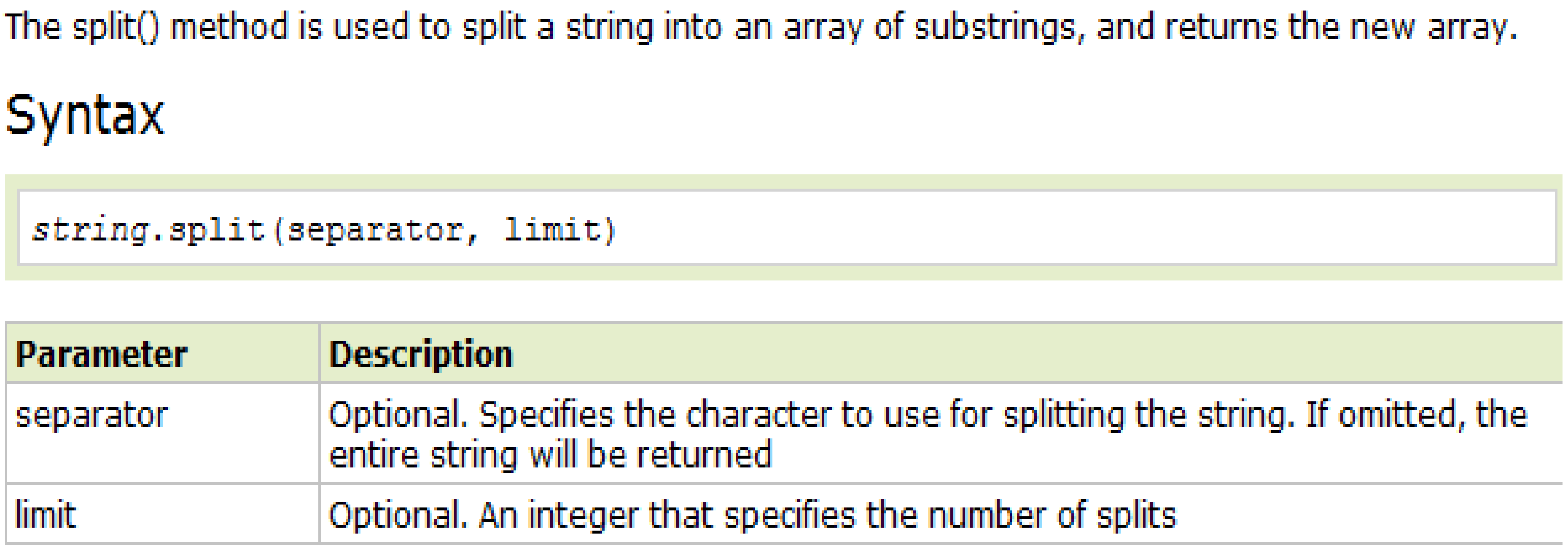
Extract different parts of a string:

```
<script type="text/javascript">

var str="Hello happy world!";

// extract all characters, start at position 0:
document.write(str.slice(0)+"<br />");

// extract all characters, start at position 6:
document.write(str.slice(6)+"<br />");

// extract from the end of the string, and to position -6:
document.write(str.slice(-6)+"<br />");

// extract only the first character:
document.write(str.slice(0,1)+"<br />");

// extract the characters from position 6 to position 11:
document.write(str.slice(6,11)+"<br />");

</script>
```

The output of the code above will be:

```
Hello happy world!
happy world!
world!
H
happy
```

split() Method



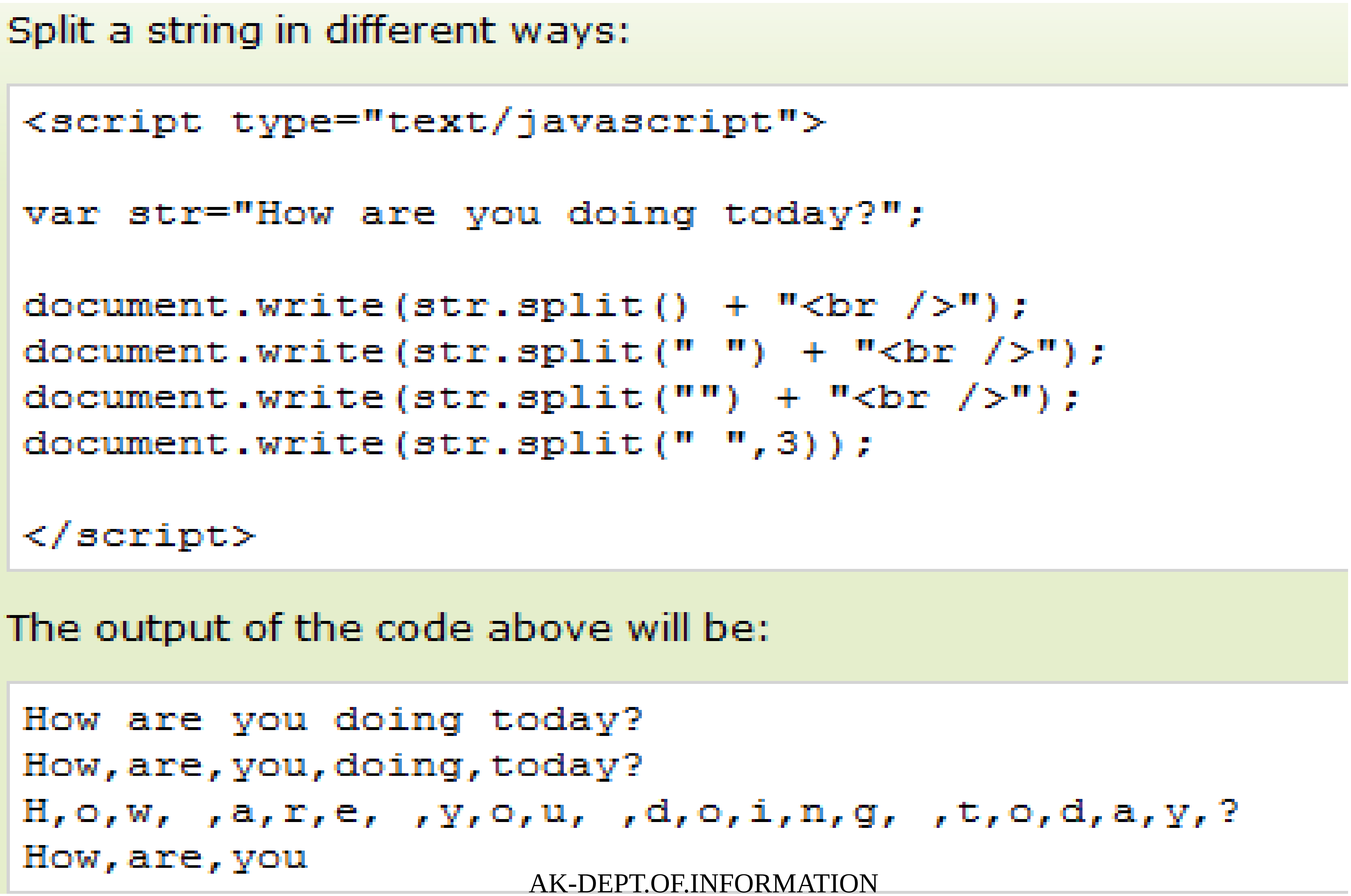
The split() method is used to split a string into an array of substrings, and returns the new array.

Syntax

```
string.split(separator, limit)
```

Parameter	Description
separator	Optional. Specifies the character to use for splitting the string. If omitted, the entire string will be returned
limit	Optional. An integer that specifies the number of splits

If an empty string ("") is used as the separator, the string is split between each character



Split a string in different ways:

```
<script type="text/javascript">

var str="How are you doing today?";

document.write(str.split() + "<br />");
document.write(str.split(" ") + "<br />");
document.write(str.split("") + "<br />");
document.write(str.split(" ",3));

</script>
```

The output of the code above will be:

```
How are you doing today?
How,are,you,doing,today?
H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?
How,are,you
```

substr() Method

The substr() method extracts the characters from a string, beginning at "start" and through the specified number of character, and returns the new sub string.

Syntax

```
string.substr(start,length)
```

Parameter	Description
start	Required. The index where to start the extraction. First character is at index 0
length	Optional. The number of characters to extract. If omitted, it extracts the rest of the string

Example

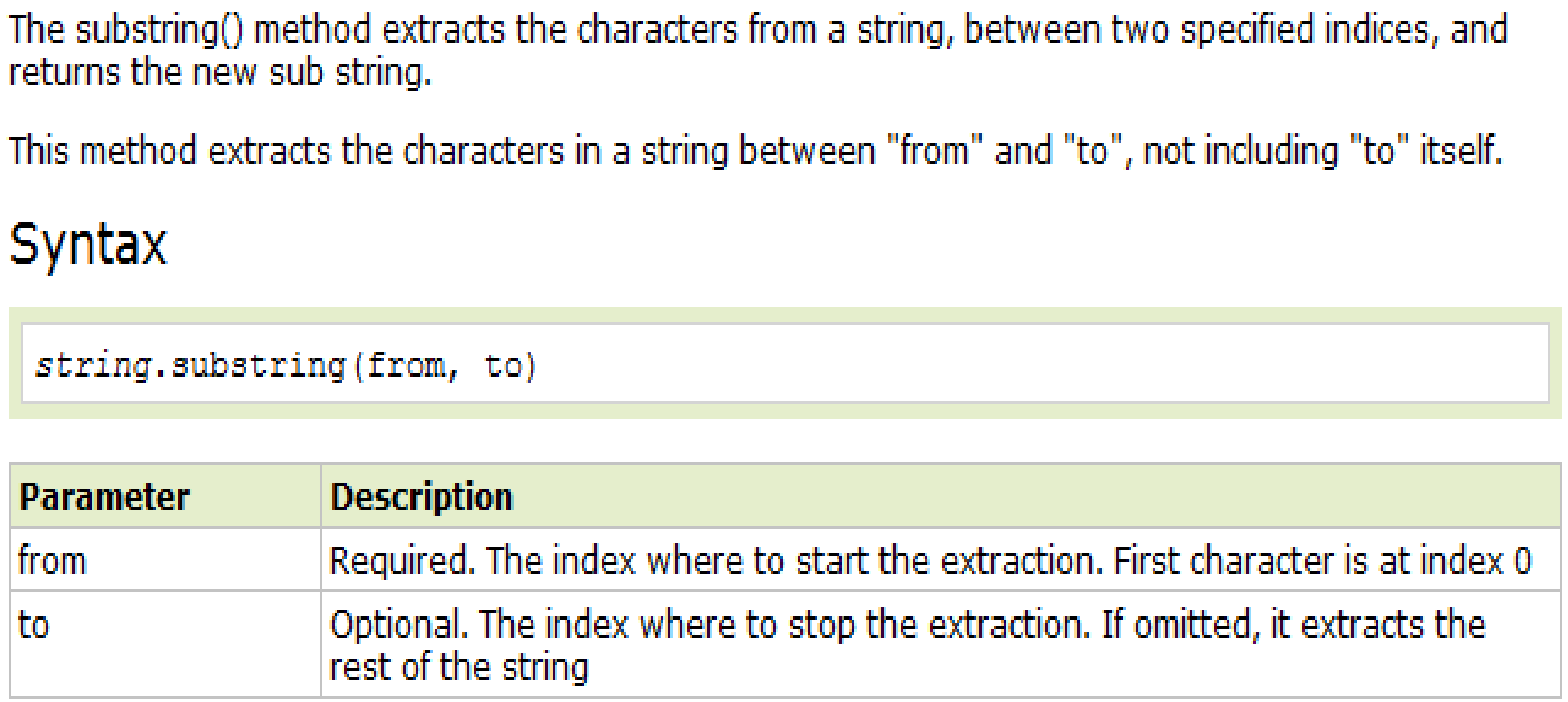
Extract characters from a string:

```
<script type="text/javascript">

var str="Hello world!";
document.write(str.substr(3)+"<br />");
document.write(str.substr(3,4));

</script>
```

The output of the code above will be:

```
lo world!
lo w
```

substring() Method



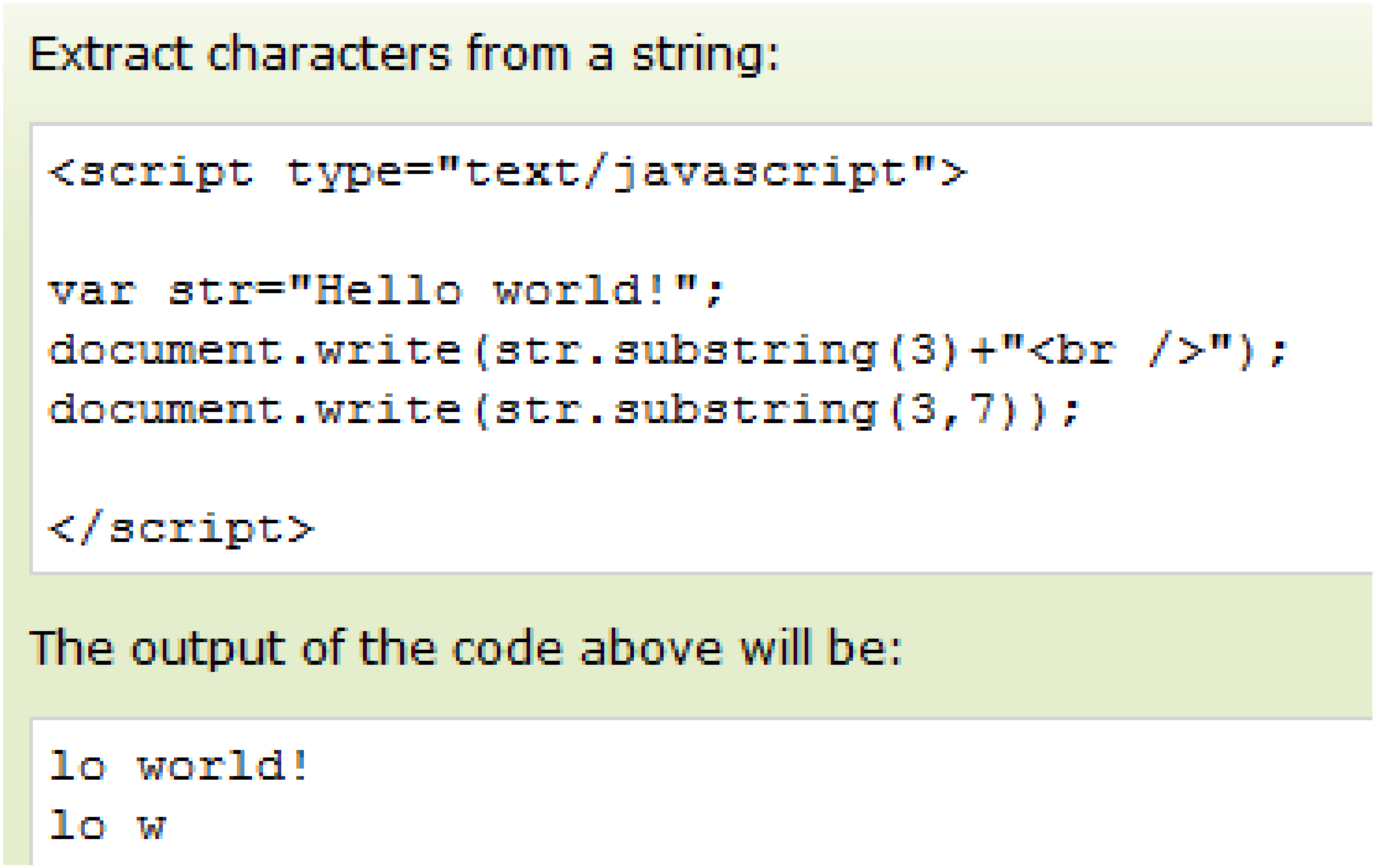
The substring() method extracts the characters from a string, between two specified indices, and returns the new sub string.

This method extracts the characters in a string between "from" and "to", not including "to" itself.

Syntax

```
string.substring(from, to)
```

Parameter	Description
from	Required. The index where to start the extraction. First character is at index 0
to	Optional. The index where to stop the extraction. If omitted, it extracts the rest of the string



Extract characters from a string:

```
<script type="text/javascript">

var str="Hello world!";
document.write(str.substring(3)+"<br />");
document.write(str.substring(3,7));

</script>
```

The output of the code above will be:

```
lo world!
lo w
```

toLowerCase() Method

The toLowerCase() method converts a string to lowercase letters.

Syntax

```
string.toLowerCase()
```

toUpperCase() Method

The toUpperCase() method converts a string to uppercase letters.
Syntax

```
string.toUpperCase()
```

```
<html>
<body>
<script type="text/javascript">
var txt="Hello World!";
document.write(txt.toLowerCase() + "<br />");
document.write(txt.toUpperCase());
</script>
</body>
</html>
```

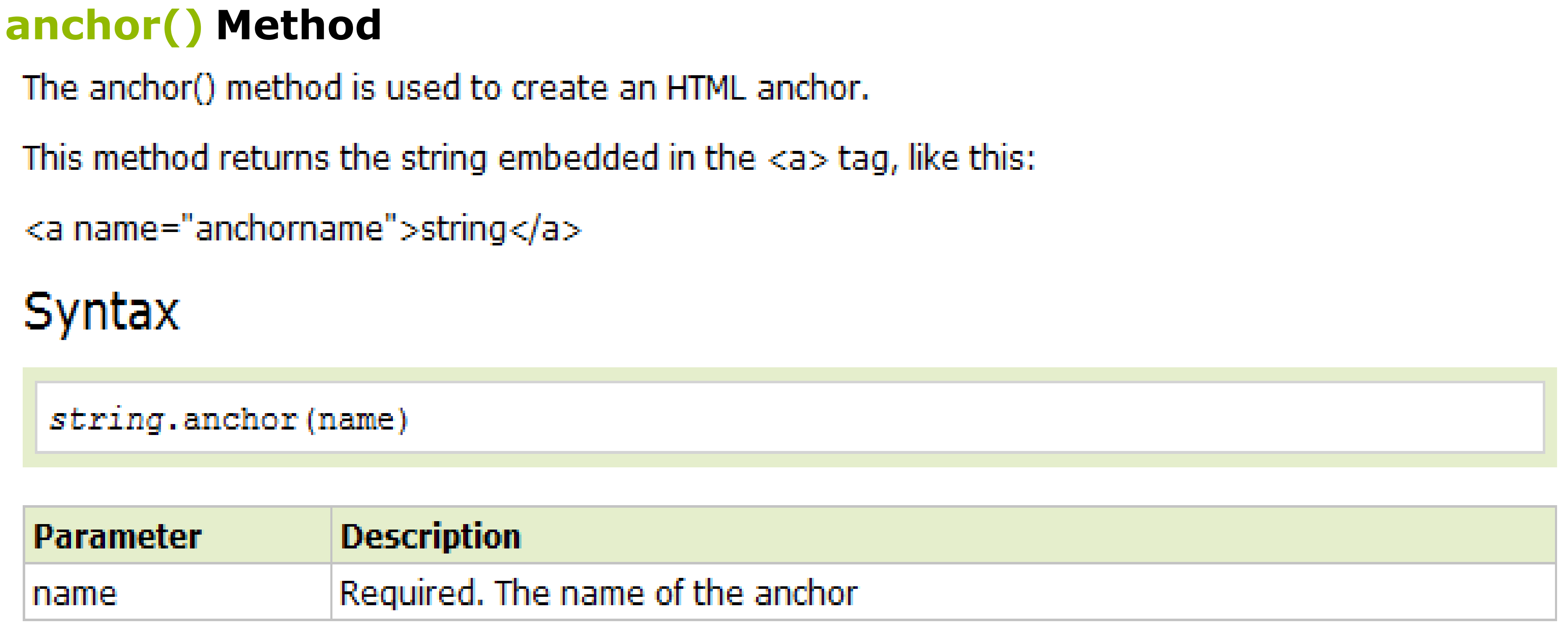
Output :

```
hello world!
HELLO WORLD!
```

String HTML Wrapper Methods

The HTML wrapper methods return the string wrapped inside the appropriate HTML tag.

Method	Description
anchor()	Creates an anchor
big()	Displays a string using a big font
blink()	Displays a blinking string
bold()	Displays a string in bold
fixed()	Displays a string using a fixed-pitch font
fontcolor()	Displays a string using a specified color
fontsize()	Displays a string using a specified size
italics()	Displays a string in italic
link()	Displays a string as a hyperlink
small()	Displays a string using a small font
strike()	Displays a string with a strikethrough
sub()	Displays a string as subscript text
sup()	Displays a string as superscript text



anchor() Method

The anchor() method is used to create an HTML anchor.

This method returns the string embedded in the <a> tag, like this:

<a name="anchorname">string</a>

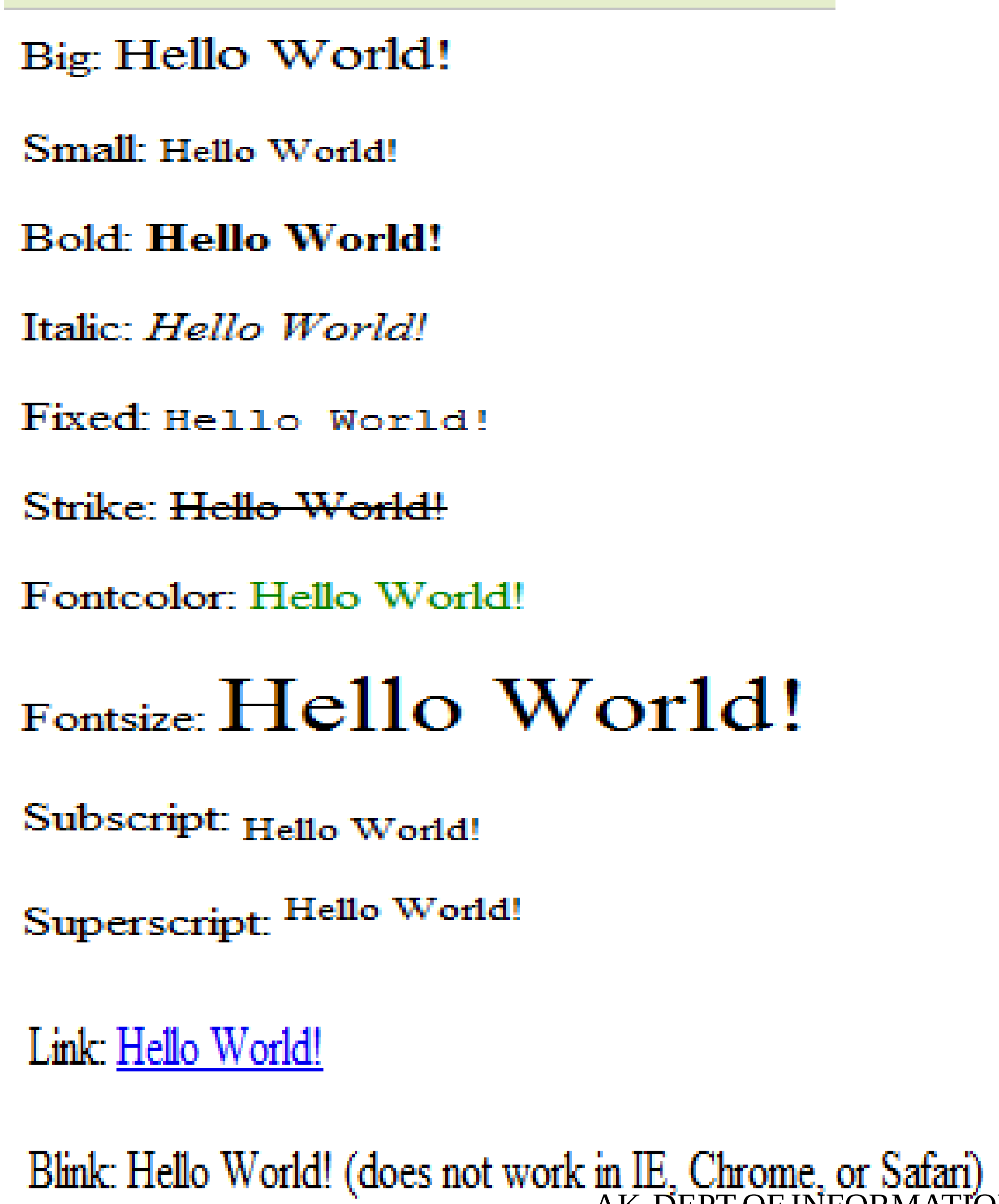
Syntax

```
string.anchor(name)
```

Parameter	Description
name	Required. The name of the anchor

```
<script type="text/javascript">

var txt = "Chapter 10";
txt.anchor("chap10");
alert(txt.anchor("myanchor"));

</script>
```

```
<html>
<body>
<script type="text/javascript">
var txt = "Hello World!";
document.write("<p>Big: " + txt.big() + "</p>");
document.write("<p>Small: " + txt.small() + "</p>");
document.write("<p>Bold: " + txt.bold() + "</p>");
document.write("<p>Italic: " + txt.italics() + "</p>");

document.write("<p>Fixed: " + txt.fixed() + "</p>");
document.write("<p>Strike: " + txt.strike() + "</p>");

document.write("<p>Fontcolor: " + txt.fontcolor("green") + "</p>");
document.write("<p>Fontsize: " + txt.fontsize(6) + "</p>");

document.write("<p>Subscript: " + txt.sub() + "</p>");
document.write("<p>Superscript: " + txt.sup() + "</p>");
document.write("<p>Link: " + txt.link("http://www.apsacollege.com") + "</p>");
document.write("<p>Blink: " + txt.blink() + " (does not work in IE, Chrome, or Safari)</p>");
</script>
</body> </html>
```



Big: Hello World!

Small: Hello World!

Bold: **Hello World!**

Italic: *Hello World!*

Fixed: `Hello World!`

Strike: ~~Hello World!~~

Fontcolor: Hello World!

Fontsize: Hello World!

Subscript: $_{\text{Hello World!}}$

Superscript: $^{\text{Hello World!}}$

Link: Hello World!

Blink: Hello World! (does not work in IE, Chrome, or Safari)

Math Object

The Math object allows you to perform mathematical tasks.

Math is not a constructor. All properties/methods of Math can be called by using Math as an object, without creating it.

Syntax

```
var x = Math.PI; // Returns PI
var y = Math.sqrt(16); // Returns the square root of 16
```

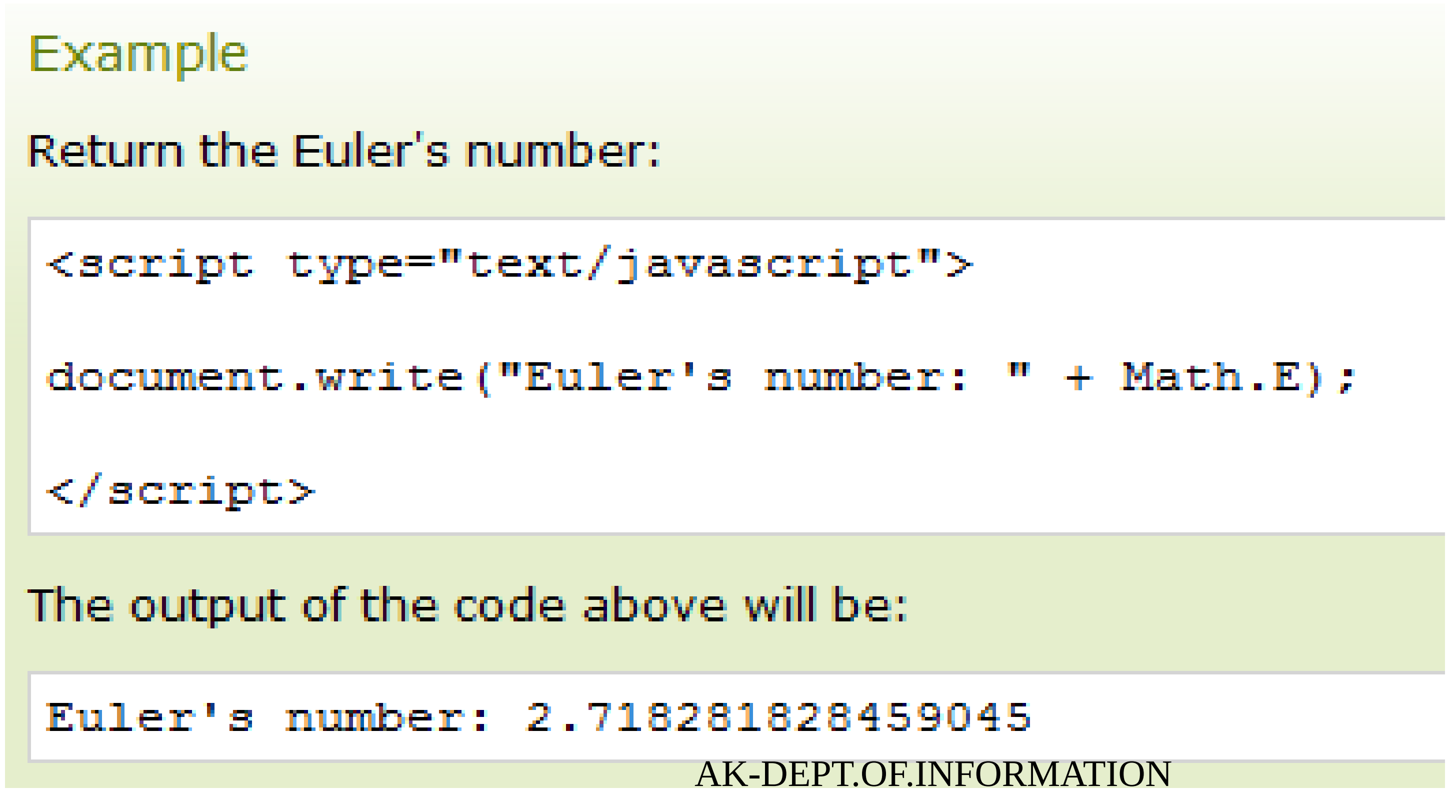
Property	Description
E	**Returns Euler's number (approx. 2.718)**
LN2	**Returns the natural logarithm of 2 (approx. 0.693)**
LN10	**Returns the natural logarithm of 10 (approx. 2.302)**
LOG2E	**Returns the base-2 logarithm of E (approx. 1.442)**
LOG10E	**Returns the base-10 logarithm of E (approx. 0.434)**
PI	**Returns PI (approx. 3.14159)**

JavaScript E Property

The E property returns the Euler's number and the base of natural logarithms, approximately 2.718.

Syntax

Math.E



Example

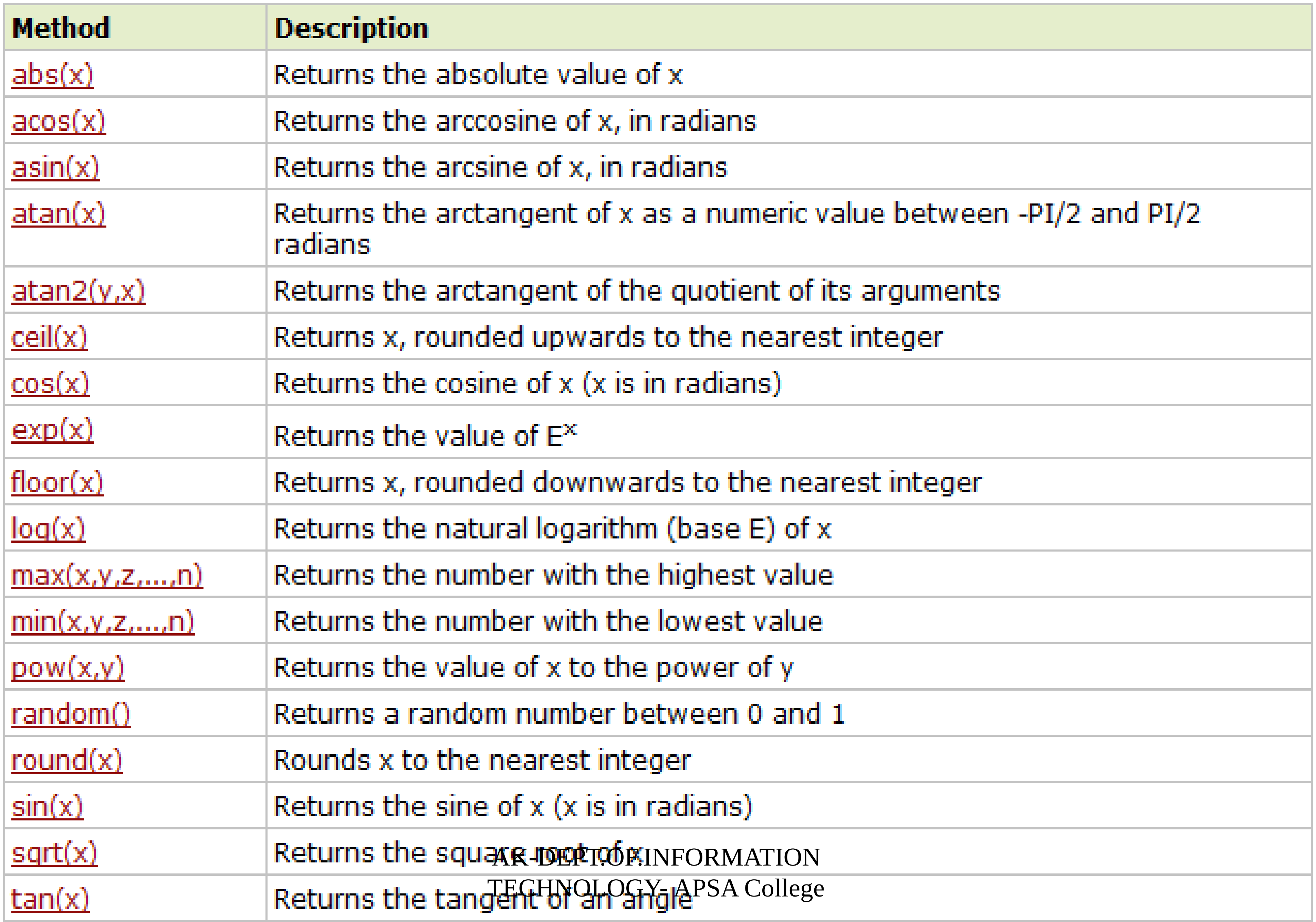
Return the Euler's number:

```
<script type="text/javascript">

document.write("Euler's number: " + Math.E);

</script>
```

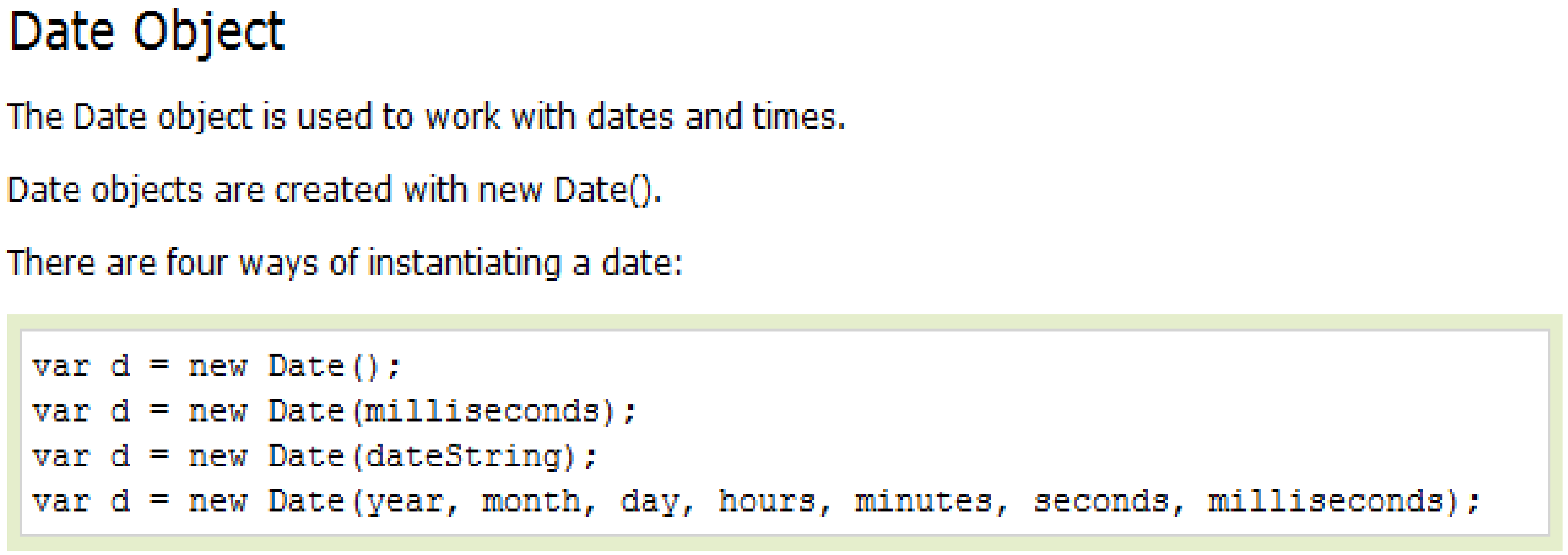
The output of the code above will be:

```
Euler's number: 2.718281828459045
```

Math Object Methods



Method	Description
abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
asin(x)	Returns the arcsine of x, in radians
atan(x)	Returns the arctangent of x as a numeric value between -PI/2 and PI/2 radians
atan2(y,x)	Returns the arctangent of the quotient of its arguments
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x (x is in radians)
exp(x)	Returns the value of E^x
floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm (base E) of x
max(x,y,z,...,n)	Returns the number with the highest value
min(x,y,z,...,n)	Returns the number with the lowest value
pow(x,y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sin(x)	Returns the sine of x (x is in radians)
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle



Date Object

The Date object is used to work with dates and times.

Date objects are created with new Date().

There are four ways of instantiating a date:

```
var d = new Date();
var d = new Date(milliseconds);
var d = new Date(dateString);
var d = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

JavaScript getDate() Method

The getDate() method returns the day of the month (from 1 to 31) for the specified date, according to local time.

Syntax

Date.getDate()

Example :

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getDate());
</script>
```

Output : 21

JavaScript getDay() Method

The getDay() method returns the day of the week (from 0 to 6) for the specified date, according to local time.

Note: Sunday is 0, Monday is 1, and so on.

Syntax

***Date*.getDay()**

Example 1

Return the day of the week:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDay());
</script>
```

output

?

Example 2

Return the name of the weekday (not just a number):

```
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
document.write("Today is " +
weekday[d.getDay()]);
</script>
```

output

```
Today is ?
```

JavaScript getMonth() Method

The getMonth() method returns the month (from 0 to 11) for the specified date, according to local time.
Note: January is 0, February is 1, and so on.

Syntax
***Date*.getMonth()**

Example 1 :Return the month:

```
<script type="text/javascript">

var d = new Date();
document.write(d.getMonth());

</script>
```

The output of the code above will be:
?

Example 2 :Return the name of the month (not just a number):

```
<script type="text/javascript">

var d=new Date();
var month=new Array(12);
month[0]="January";
month[1]="February";
month[2]="March";
month[3]="April";
month[4]="May";
month[5]="June";
month[6]="July";
month[7]="August";
month[8]="September";
month[9]="October";
month[10]="November";
month[11]="December";

document.write("The current month is " + month[d.getMonth()]);

</script>
```

The output of the code above will be:

```
The current month is ?
```

JavaScript getFullYear() Method

The getFullYear() method returns the year (four digits) of the specified date, according to local time.

Syntax

***Date*.getFullYear()**

Example 1 :Return the four-digit year:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getFullYear());
</script>
```

output : `2011`

Example 2:Return the four-digit year from a specific date:

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write("I was born in " + d.getFullYear());
</script>
```

output :

```
I was born in 1983
```

JavaScript **getTime()** Method

The getTime() method returns the number of milliseconds since midnight of January 1, 1970 and the specified date.

Syntax

***Date*.getTime()**

Example 1 :Return the number of milliseconds since 1970/01/01:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getTime() + "milliseconds since 1970/01/01");
</script>
```

Example 2 : Calculate the number of years since 1970/01/01:

```
<script type="text/javascript">

var minutes=1000*60;
var hours=minutes*60;
var days=hours*24;
var years=days*365;
var d=new Date();
var t=d.getTime();
var y=t/years;
document.write("It's been" + Math.round(y) + "years since 1970/01/01!");
</script>
```

JavaScript getHours() Method

The getHours() method returns the hour (from 0 to 23) of the specified date and time, according to local time.

Syntax

Date.getHours()

Example 1:Return the hour, according to local time:

```
<script type="text/javascript">

var d = new Date();
document.write(d.getHours());
</script>
```

output :

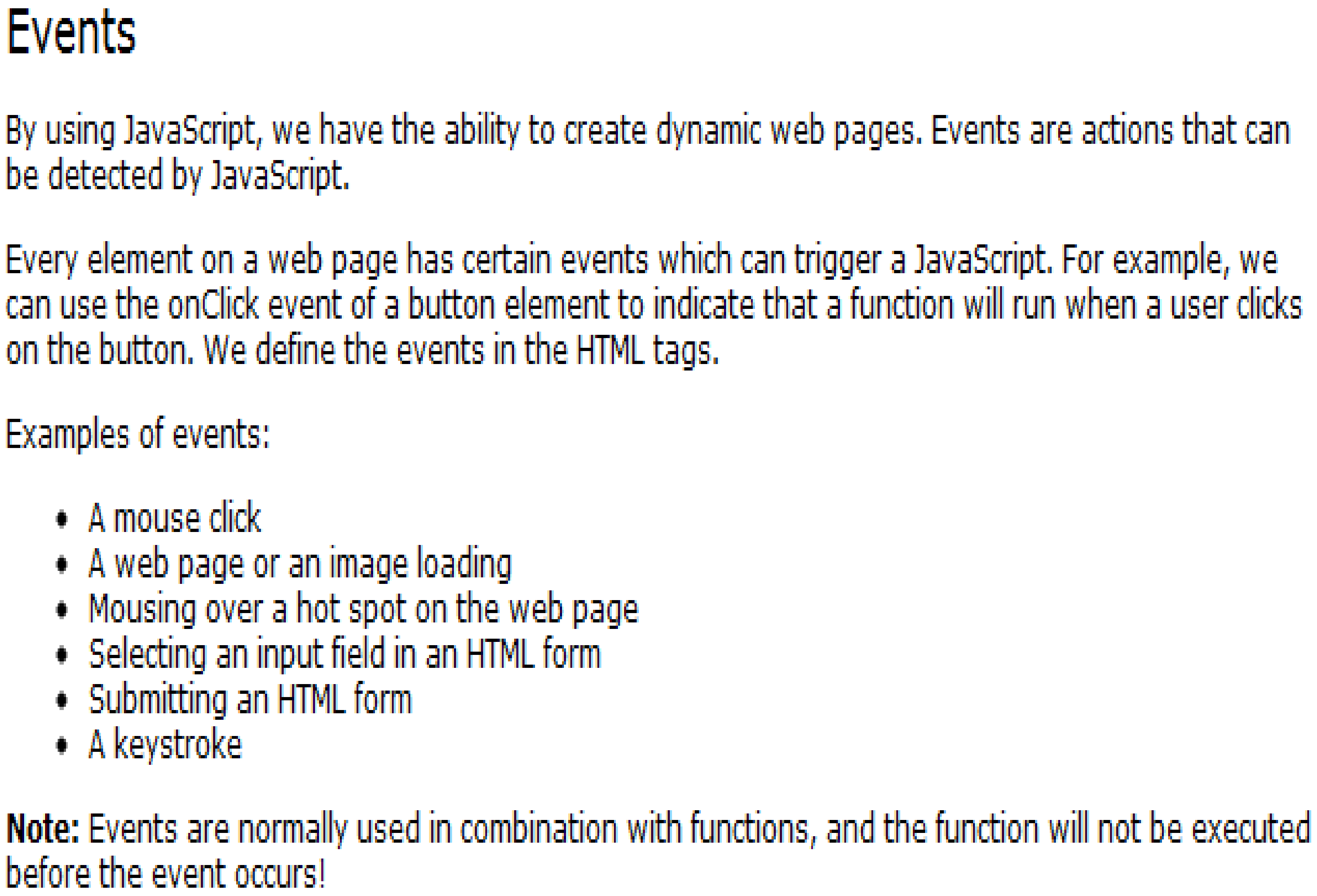
?

Example 2: Return the hour from a specific date and time:

```
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getHours());
</script>
```

output :

1



Events

By using JavaScript, we have the ability to create dynamic web pages. Events are actions that can be detected by JavaScript.

Every element on a web page has certain events which can trigger a JavaScript. For example, we can use the onClick event of a button element to indicate that a function will run when a user clicks on the button. We define the events in the HTML tags.

Examples of events:

- A mouse click
- A web page or an image loading
- Mousing over a hot spot on the web page
- Selecting an input field in an HTML form
- Submitting an HTML form
- A keystroke

Note: Events are normally used in combination with functions, and the function will not be executed before the event occurs!

onLoad and onUnload

The onLoad and onUnload events are triggered when the user enters or leaves the page.

The onLoad event is often used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

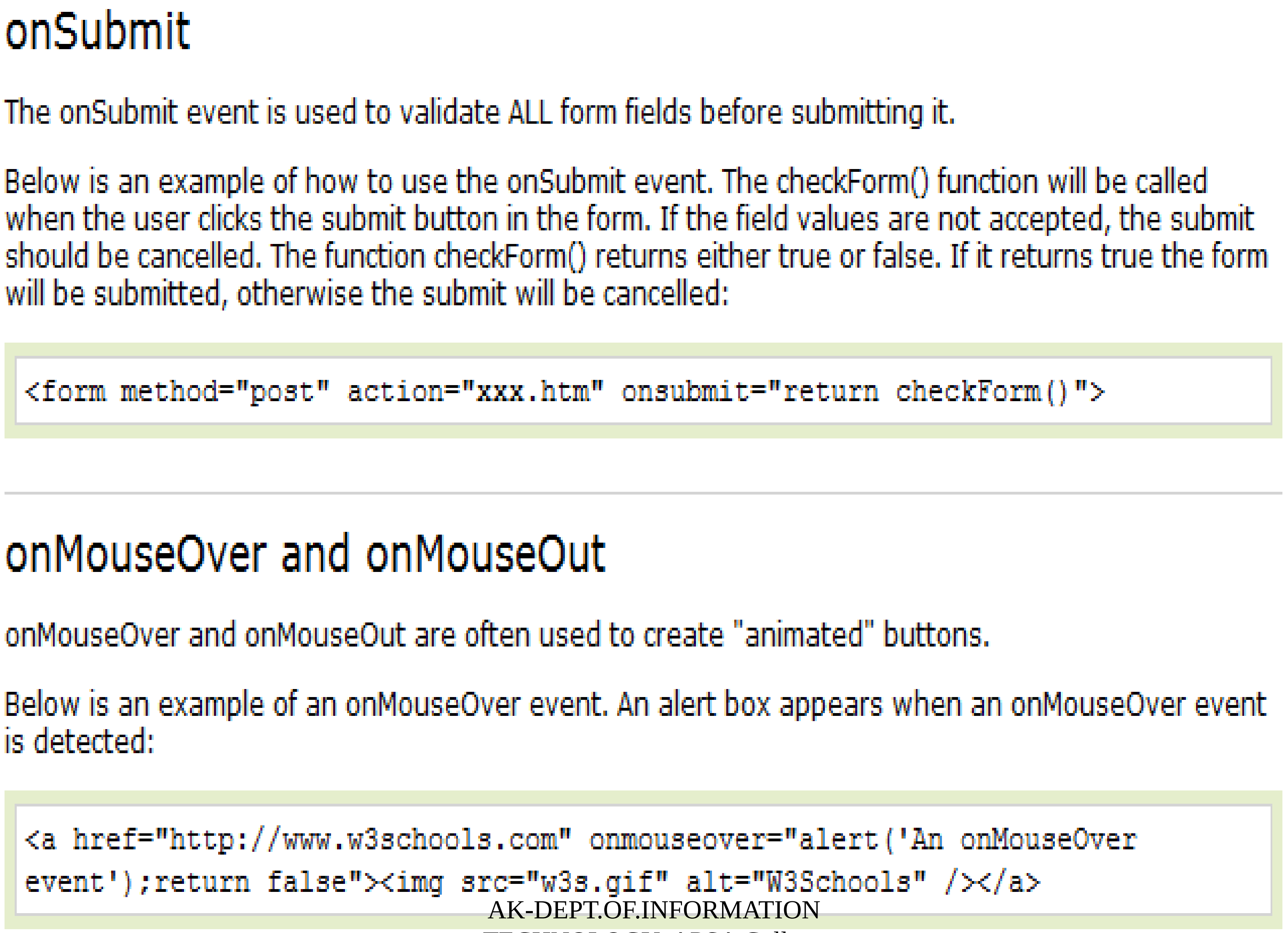
Both the onLoad and onUnload events are also often used to deal with cookies that should be set when a user enters or leaves a page. For example, you could have a popup asking for the user's name upon his first arrival to your page. The name is then stored in a cookie. Next time the visitor arrives at your page, you could have another popup saying something like: "Welcome John Doe!".

onFocus, onBlur and onChange

The onFocus, onBlur and onChange events are often used in combination with validation of form fields.

Below is an example of how to use the onChange event. The checkEmail() function will be called whenever the user changes the content of the field:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```



onSubmit

The onSubmit event is used to validate ALL form fields before submitting it.

Below is an example of how to use the onSubmit event. The checkForm() function will be called when the user clicks the submit button in the form. If the field values are not accepted, the submit should be cancelled. The function checkForm() returns either true or false. If it returns true the form will be submitted, otherwise the submit will be cancelled:

```
<form method="post" action="xxx.htm" onsubmit="return checkForm()">
```

onMouseOver and onMouseOut

onMouseOver and onMouseOut are often used to create "animated" buttons.

Below is an example of an onMouseOver event. An alert box appears when an onMouseOver event is detected:

```
<a href="http://www.w3schools.com" onmouseover="alert('An onMouseOver
event');return false"><img src="w3s.gif" alt="W3Schools" /></a>
```

Event	Value	Description
onchange	**script**	**Script runs when the element changes**
onsubmit	**script**	**Script runs when the form is submitted**
onreset	**script**	**Script runs when the form is reset**
onselect	**script**	**Script runs when the element is selected**
onblur	**script**	**Script runs when the element loses focus**
onfocus	**script**	**Script runs when the element gets focus**
onkeydown	**script**	**Script runs when key is pressed**
onkeypress	**script**	**Script runs when key is pressed and released**
onkeyup	**script**	**Script runs when key is released**
onclick	**script**	**Script runs when a mouse click**
ondblclick	**script**	**Script runs when a mouse double-click**
onmousedown	**script**	**Script runs when mouse button is pressed**
onmousemove	**script**	**Script runs when mouse pointer moves**
onmouseout	**script**	**Script runs when mouse pointer moves out of an element**
onmouseover	**script**	**Script runs when mouse pointer moves over an element**
onmouseup	**script**	**Script runs when mouse button is released**

Dynamic HTML: Filters and Transitions

Outline

Objectives

- To use filters to achieve special effects.
- To combine filters to achieve an even greater variety of special effects.
- To be able to create animated visual transitions between Web pages.
- To be able to modify filters dynamically, using DHTML.

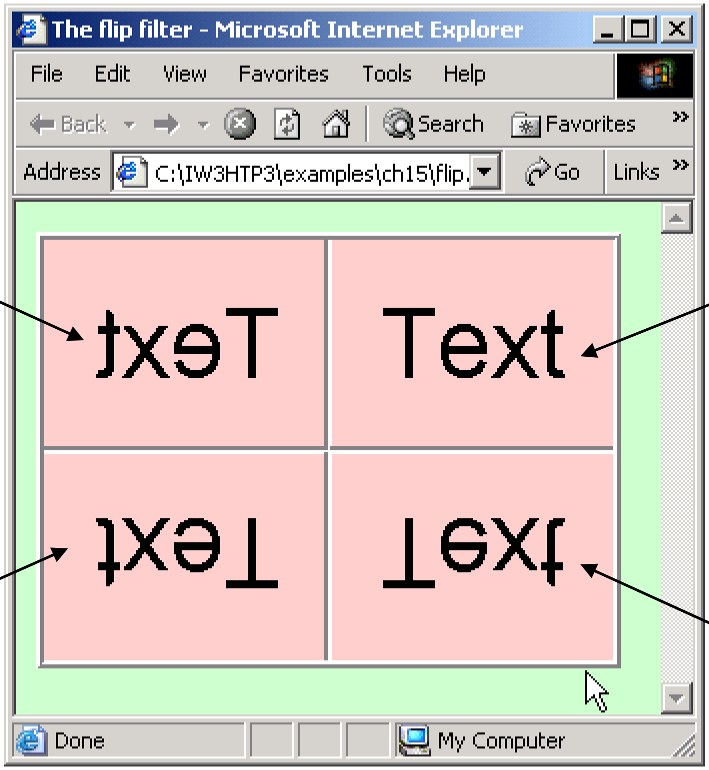
Introduction

- Filters
  - Cause changes that are persistent
- Transitions
  - Temporary
  - Allow transfer from one page to another with pleasant visual effect
    - For example, random dissolve

Flip Filters: `flipv` and `fliph`

- `flipv` and `fliph` filters mirror text or images vertically and horizontally, respectively

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig. 15.1: flip.html   -->
<!-- Using the flip filters -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>The flip filter</title>

      <style type = "text/css">
         body  { background-color: #CCFFCC }

         table { font-size: 3em;
                 font-family: Arial, sans-serif;
                 background-color: #FFCCCC;
                 border-style: ridge ;
                 border-collapse: collapse }

         td    { border-style: groove;
                 padding: 1ex }
      </style>
    </head>

   <body>

      <table>

         <tr>
            <!-- Filters are applied in style declarations -->
            <td style = "filter: fliph">Text</td>
            <td>Text</td>
         </tr>

         <tr>
            <!-- More than one filter can be applied at once -->
            <td style = "filter: flipv fliph">Text</td>
            <td style = "filter: flipv">Text</td>
         </tr>

      </table>

   </body>
</html>
```


The flip filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\flip.
Go Links
Text
Text
Text
Text
Done
My Computer

`fliph` filter applied

No filters applied

Both `fliph` and `flipv` filters applied

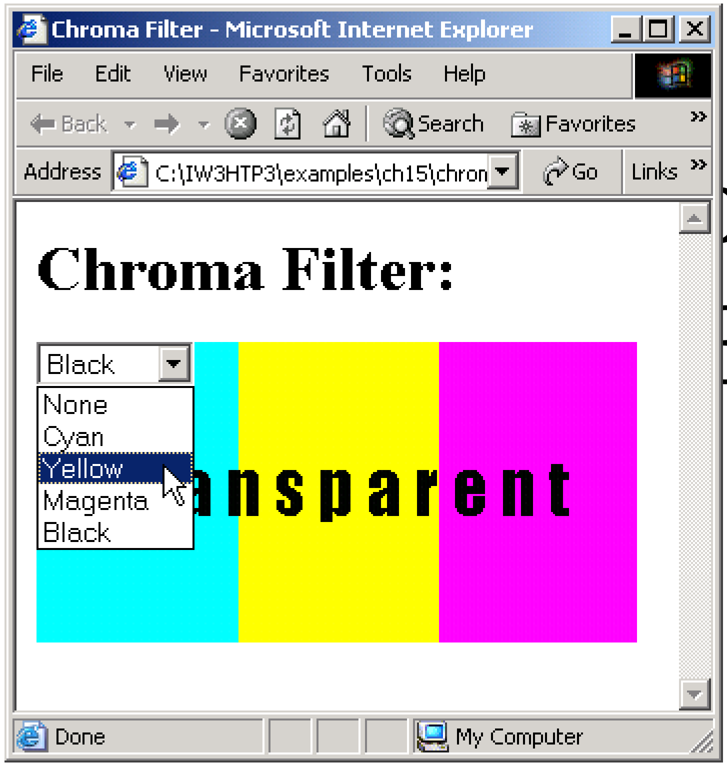
`flipv` filter applied

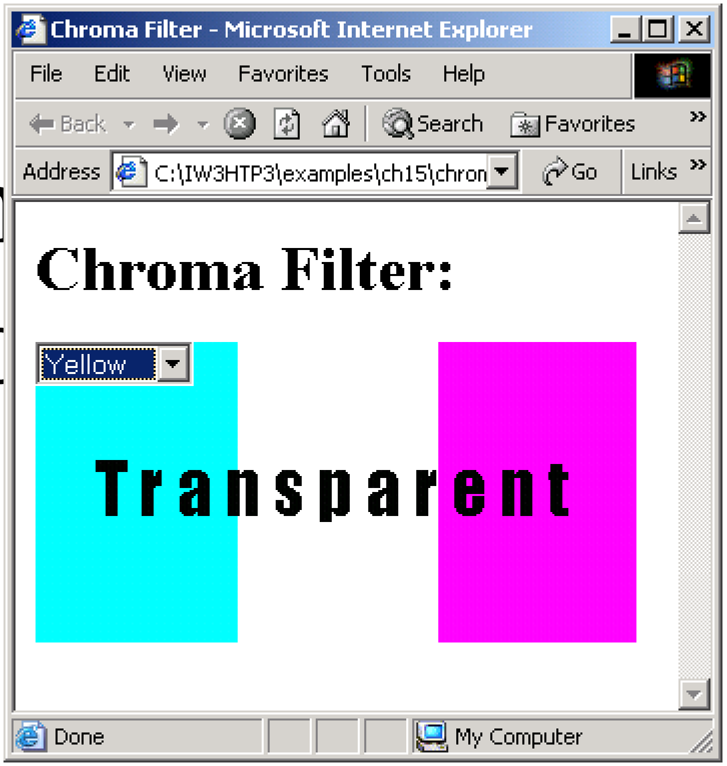
Transparency with the `chroma` Filter

- `chroma` filter applies transparency effects dynamically
  - Without using a graphics editor to hard-code transparency into the image
- `onchange`
  - Fires when the `value` of a form changes

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.2: chroma.html                              -->
<!-- Applying transparency using the chroma filter    -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Chroma Filter</title>

      <script type = "text/javascript">
         <!--
         function changecolor( theColor )
         {
            if ( theColor ) {
               // if the user selected a color, parse the
               // value to hex and set the filter color.
               chromaImg.filters( "chroma" ).color = theColor;
               chromaImg.filters( "chroma" ).enabled = true;
            }
            else // if the user selected "None",
               // disable the filter.
               chromaImg.filters( "chroma" ).enabled = false;
         }
      // -->
      </script>
   </head>

   <body>

      <h1>Chroma Filter:</h1>

      <img id = "chromaImg" src = "trans.gif" style =
         "position: absolute; filter: chroma"  alt =
         "Transparent Image" />

      <form action = "">
         <!-- The onchange event fires when -->
         <!-- a selection is changed        -->
         <select onchange = "changecolor( this.value )">
            <option value = "">None</option>
            <option value = "#00FFFF">Cyan</option>
            <option value = "#FFFF00">Yellow</option>
            <option value = "#FF00FF">Magenta</option>
            <option value = "#000000" selected = "selected">
               Black</option>
         </select>
      </form>

```

```
    </body>
</html>
```

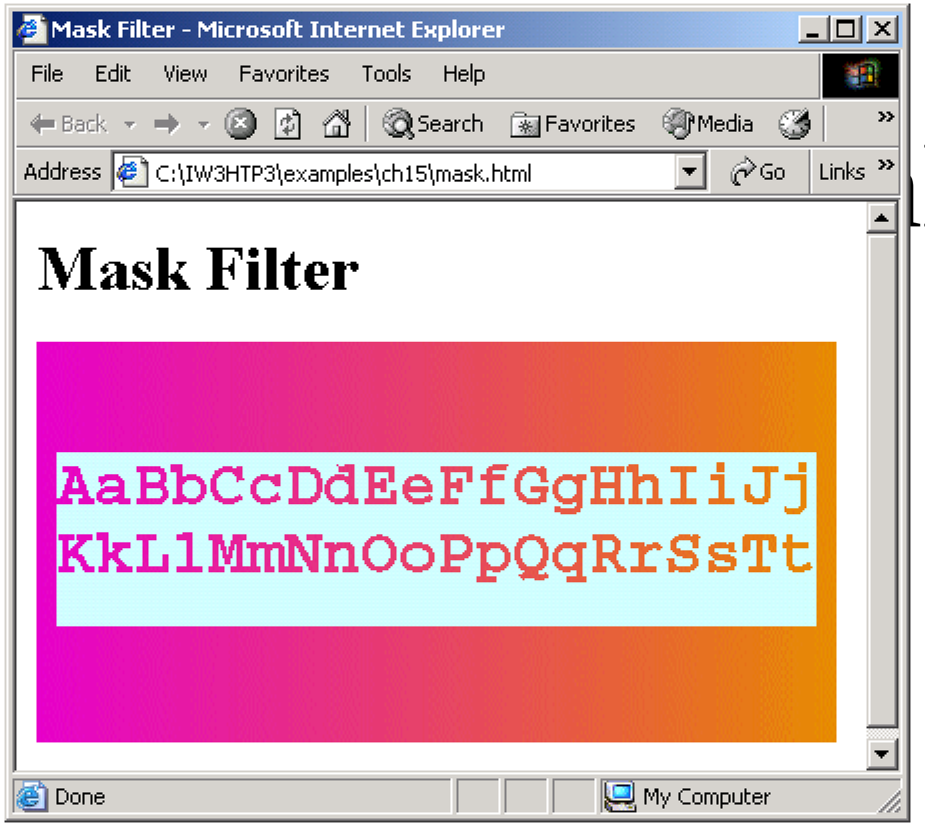

Chroma Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\chron Go Links
Chroma Filter:
Black
None
Cyan
Yellow
Magenta
Black
ansparent
Done
My Computer


Chroma Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\chron Go Links
Chroma Filter:
Yellow
Transparent
Done
My Computer

Creating Image masks

- Background is a solid color
- Foreground is transparent to the image or color behind it

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.3: mask.html          -->
<!-- Placing a mask over an image -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Mask Filter</title>
    </head>

    <body>

      <h1>Mask Filter</h1>

      <!-- Filter parameters are specified in parentheses, -->
      <!-- in the form param1 = value1, param2 = value2,   -->
      <!-- etc.                                            -->
      <div style = "position: absolute; top: 125; left: 20;
         filter: mask( color = #CCFFFF )">
      <h1 style = "font-family: Courier, monospace">
      AaBbCcDdEeFfGgHhIiJj<br />
      KkLlMmNnOoPpQqRrSsTt
      </h1>
      </div>

      <img src = "gradient.gif" width = "400" height = "200"
         alt = "Image with Gradient Effect" />
   </body>
</html>
```


Mask Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\mask.html
Go Links
Done
My Computer

Mask Filter

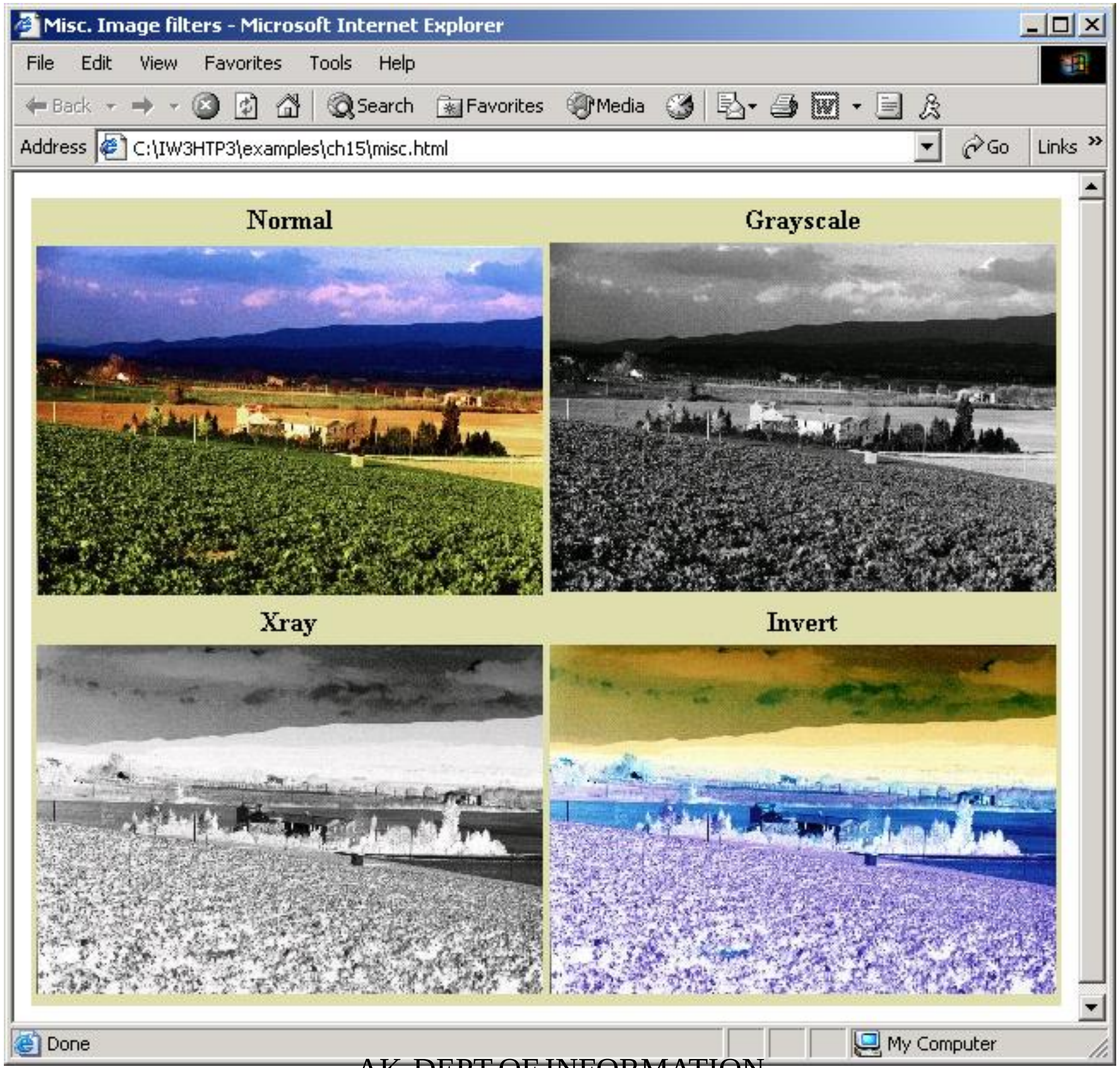
AaBbCcDdEeFfGgHhIiJj
KkLlMmNnOoPpQqRrSsTt

Miscellaneous Image Filters: `invert`, `gray` and `xray`

- `invert` filter
  - Negative image effect
    - Dark areas become light and light areas become dark
- `gray` filter
  - Grayscale image effect
    - All color is stripped from the image, only brightness data remains
- `xray` filter
  - X-ray effect
    - Inversion of the grayscale effect

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig 15.4: misc.html                                   -->
<!-- Image filters to invert, grayscale or xray an image -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Misc. Image filters</title>

      <style type = "text/css">
         .cap { font-weight: bold;
                background-color: #DDDDAA;
                text-align: center }
      </style>
    </head>

    <body>
      <table class = "cap">
         <tr>
            <td>Normal</td>
            <td>Grayscale</td>
         </tr>
         <tr>
```

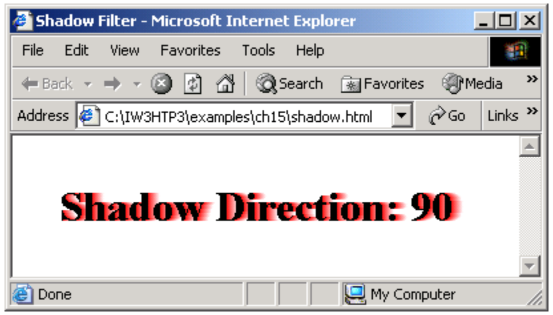
```
               <td><img src = "hc.jpg" alt =
                  "normal scenic view" /></td>
               <td><img src = "hc.jpg" style = "filter: gray"
                     alt = "gray scenic view"/>
               </td>
            </tr>
            <tr>
               <td>Xray</td>
               <td>Invert</td>
            </tr>
            <tr>
               <td><img src = "hc.jpg" style = "filter: xray"
                     alt = "xray scenic view"/>
               </td>
               <td><img src = "hc.jpg" style = "filter: invert"
                  alt = "inverted scenic view"/>
               </td>
            </tr>
         </table>

      </body>
   </html>
```


Misc. Image filters - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\misc.html
Go Links
Normal
Grayscale
Xray
Invert
Done
My Computer

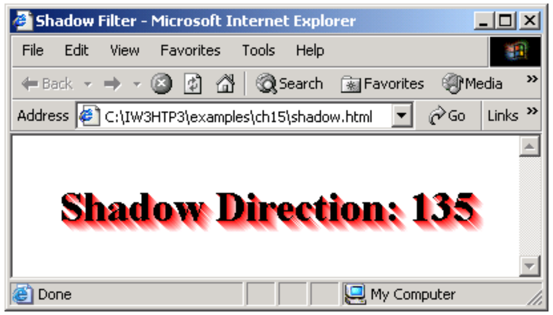
Adding `shadows` to Text

- `shadow` filter
  - Showing effect
    - Three-dimensional appearance

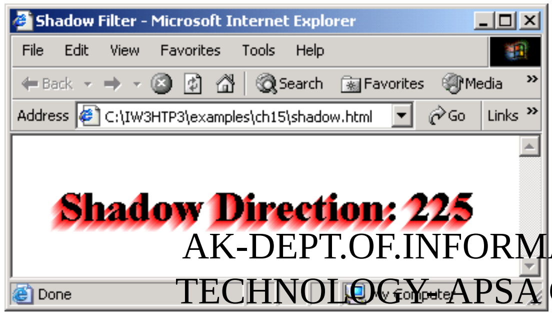
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig 15.5: shadow.html      -->
<!-- Applying the shadow filter -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Shadow Filter</title>

      <script type = "text/javascript">
         <!--
         var shadowDirection = 0;

         function start()
         {
            window.setInterval( "runDemo()", 500 );
         }

         function runDemo()
         {
            shadowText.innerText =
               "Shadow Direction: " + shadowDirection % 360;
            shadowText.filters( "shadow" ).direction =
               ( shadowDirection % 360 );
            shadowDirection += 45;
         }
         // -->
      </script>
   </head>

   <body onload = "start()">

      <h1 id = "shadowText" style = "position: absolute; top: 25;
         left: 25; padding: 10; filter: shadow( direction = 0,
         color = red )">Shadow Direction: 0</h1>
   </body>
</html>
```


Shadow Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\shadow.html Go Links
Done My Computer

Shadow Direction: 90


Shadow Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\shadow.html Go Links
Done My Computer

Shadow Direction: 135


Shadow Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\shadow.html Go Links
Done My Computer

Shadow Direction: 225

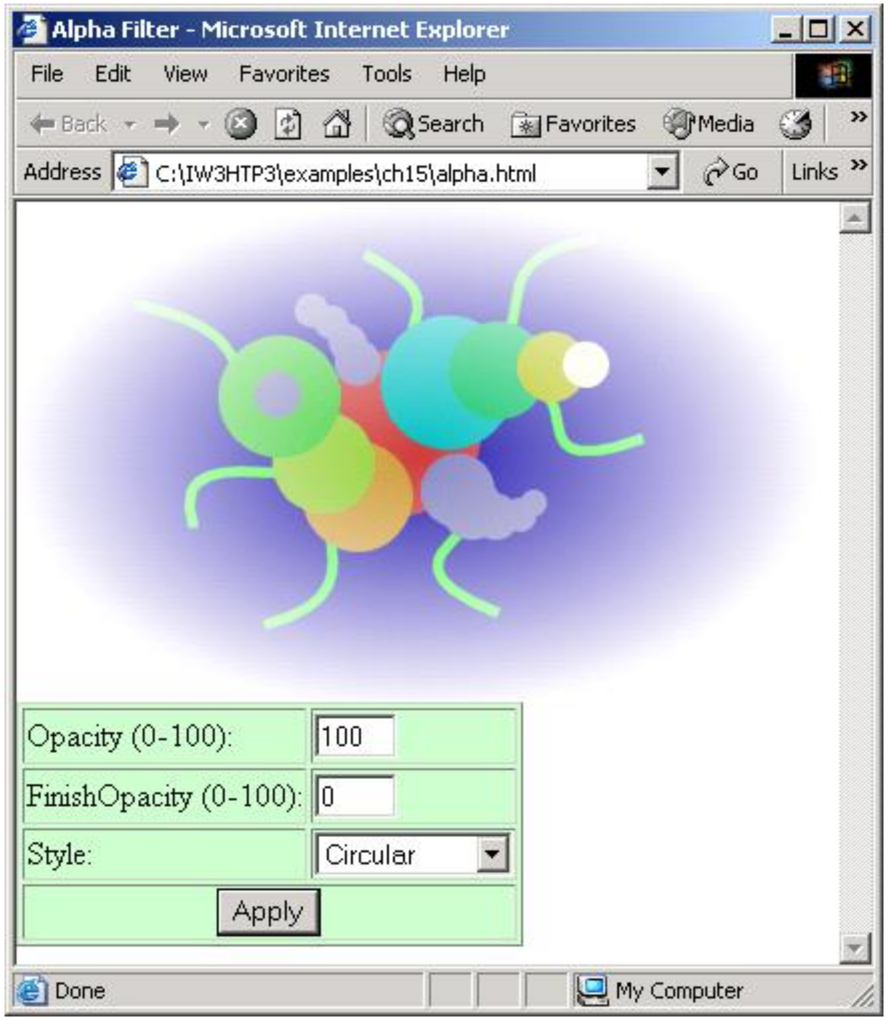
Creating Gradients with `alpha`

- `alpha` filter
  - Gradient effect
    - Gradual progression from starting color to target color
  - `style`
    - Uniform opacity (`value` 0)
    - Linear gradient (`value` 1)
    - Circular gradient (`value` 2)
    - Rectangular gradient (`value` 3)

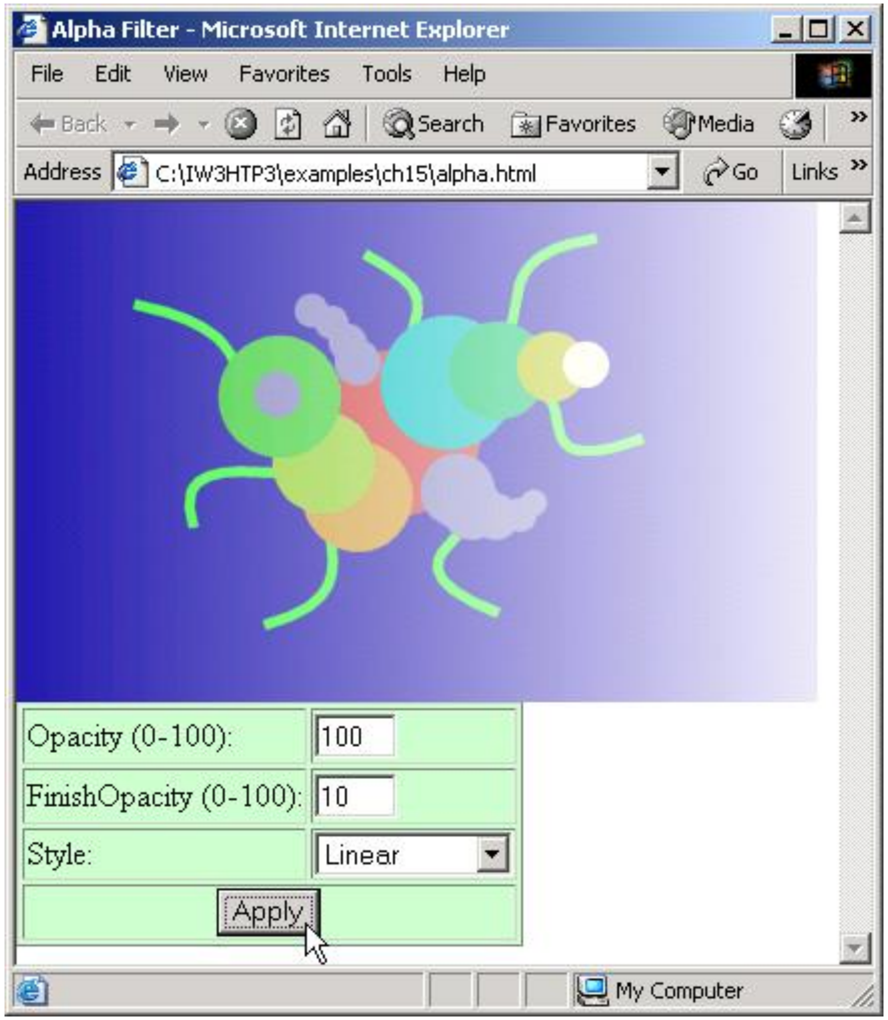
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig 15.6: alpha.html                  -->
<!-- Applying the alpha filter to an image -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Alpha Filter</title>
    <script type = "text/javascript">
      <!--
      function run()
      {
         pic.filters( "alpha" ).opacity = opacityButton.value;
         pic.filters( "alpha" ).finishopacity =
            opacityButton2.value;
         pic.filters( "alpha" ).style = styleSelect.value;
      }
      // -->
    </script>
    </head>

    <body>

```

```
      <div id = "pic"
           style = "position: absolute; left:0; top: 0;
                    filter: alpha( style = 2, opacity = 100,
                    finishopacity = 0 )">
         <img src = "flag.gif" alt = "Flag" />
      </div>

      <table style = "position: absolute; top: 250; left: 0;
         background-color: #CCFFCC" border = "1">

         <tr>
            <td>Opacity (0-100):</td>
            <td><input type = "text" id = "opacityButton"
               size = "3" maxlength = "3" value = "100" /></td>
         </tr>

         <tr>
           <td>FinishOpacity (0-100):</td>
            <td><input type = "text" id = "opacityButton2"
               size = "3" maxlength = "3" value = "0" /></td>
         </tr>

         <tr>
            <td>Style:</td>
            <td><select id = "styleSelect">
```

```
                  <option value = "1">Linear</option>
                  <option value = "2" selected = "selected">
                     Circular</option>
                  <option value = "3">Rectangular</option>
                  </select></td>
            </tr>

            <tr>
               <td align = "center" colspan = "2">
                  <input type = "button" value = "Apply"
                     onclick = "run()" />
               </td>
            </tr>
         </table>

      </body>
   </html>
```


Alpha Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\alpha.html Go Links
Done
My Computer

Opacity (0-100):	100
FinishOpacity (0-100):	0
Style:	Circular
Apply	

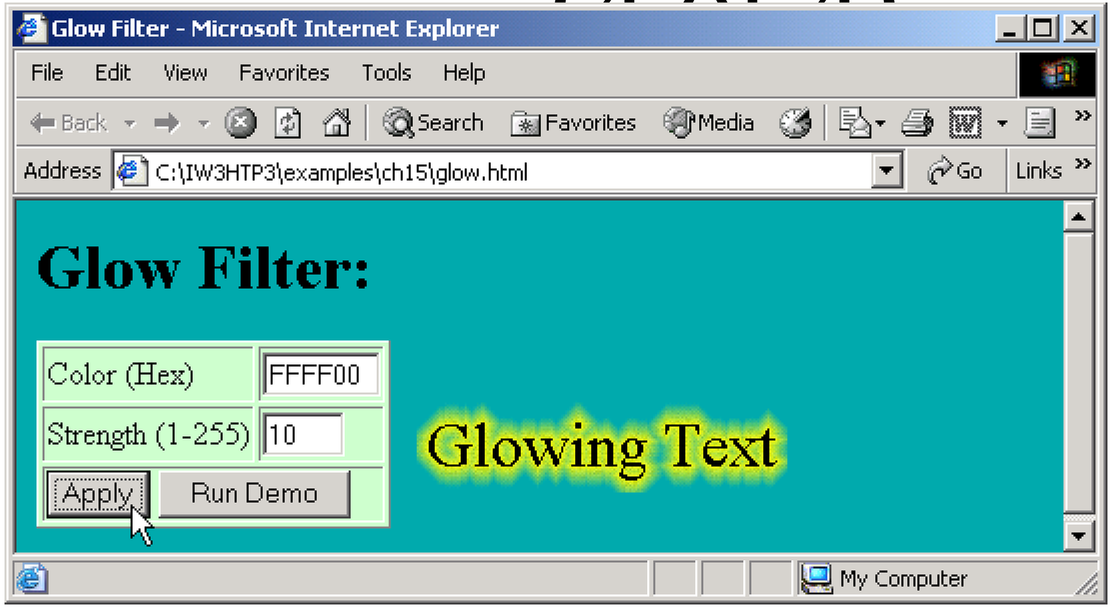

Alpha Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\alpha.html Go Links
My Computer

Opacity (0-100):	100
FinishOpacity (0-100):	10
Style:	Linear
Apply	

Making Text `glow`

- `glow` filter adds an aura of color around text

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.7: glow.html      -->
<!-- Applying the glow filter -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Glow Filter</title>
      <script type = "text/javascript">
         <!--
         var strengthIndex = 1;
         var counter = 1;
         var upDown = true;
         var colorArray = [ "FF0000", "FFFF00", "00FF00",
                            "00FFFF", "0000FF", "FF00FF" ];
         function apply()
         {
            glowSpan.filters( "glow" ).color =
               parseInt( glowColor.value, 16 );
            glowSpan.filters( "glow" ).strength =
               glowStrength.value;
         }

         function startdemo()
         {
            window.setInterval( "rundemo()", 150 );
         }

         function rundemo()
         {
            if ( upDown ) {
               glowSpan.filters( "glow" ).strength =
                  strengthIndex++;
            }
            else {
               glowSpan.filters( "glow" ).strength =
                  strengthIndex--;
            }

            if ( strengthIndex == 1 ) {
               upDown = !upDown;
               counter++;
               glowSpan.filters( "glow" ).color =
                  parseInt( colorArray[ counter % 6 ], 16 );
            }

            if ( strengthIndex == 10 ) {
               upDown = !upDown;
            }
         }
         // -->
      </script>
   </head>

   <body style = "background-color: #00AAAA">
      <h1>Glow Filter:</h1>

      <span id = "glowSpan" style = "position: absolute;
         left: 200;top: 100; padding: 5; filter: glow(
         color = red, strength = 5 ); font-size: 2em">
         Glowing Text
      </span>

      <table border = "1" style = "background-color: #CCFFCC">
         <tr>
            <td>Color (Hex)</td>
            <td><input id = "glowColor" type = "text" size = "6"
               maxlength = "6" value = "FF0000" /></td>
         </tr>
         <tr>
            <td>Strength (1-255)</td>
            <td><input id = "glowStrength" type = "text"
                   size = "3" maxlength = "3" value = "5" />
            </td>
         </tr>
         <tr>
            <td colspan = "2">
               <input type = "button" value = "Apply"
                  onclick = "apply()" />
               <input type = "button" value = "Run Demo"
                  onclick = "startdemo()" /></td>
         </tr>
      </table>

   </body>
</html>
```


Glow Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\glow.html
Go Links
Apply
Run Demo
My Computer

Glow Filter:

Color (Hex) FFFF00
Strength (1-255) 10

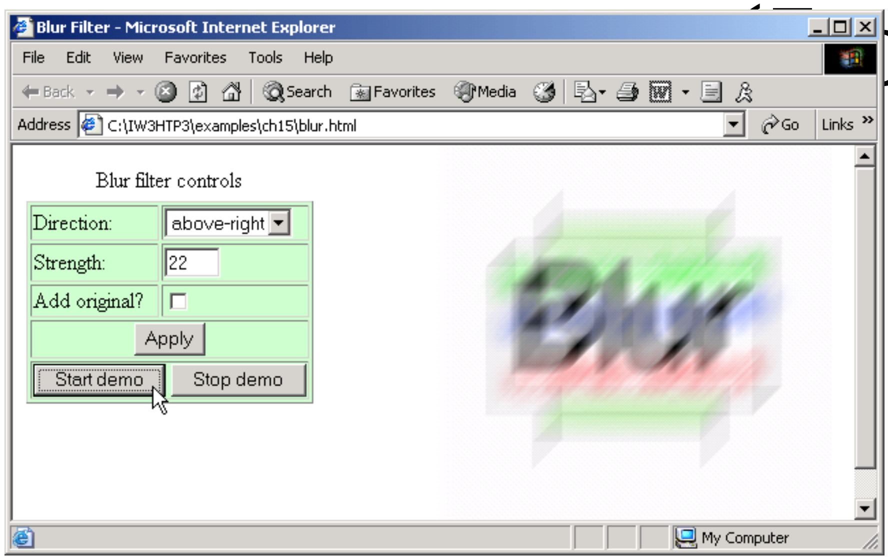
Glowing Text

Creating Motion with `blur`

- `blur` filter creates an illusion of motion by blurring text or images in a certain direction
  - `Add`
    - Adds a copy of the original image over the blurred image
  - `Direction`
    - Determines in which direction the `blur` filter is applied
  - `strength`
    - Determines how strong the blurring effect is

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.8: blur.html -->
<!-- The blur filter    -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Blur Filter</title>
      <script type = "text/javascript">
         <!--
         var strengthIndex = 1;
         var blurDirection = 0;
         var upDown = 0;
         var timer;

         function reBlur()
         {
            blurImage.filters( "blur" ).direction =
               document.forms( "myForm" ).Direction.value;
            blurImage.filters( "blur" ).strength =
               document.forms( "myForm" ).Strength.value;
            blurImage.filters( "blur" ).add =
               document.forms( "myForm" ).AddBox.checked;
         }

         function startDemo()
         {
            timer = window.setInterval( "runDemo()", 5 );
         }

         function runDemo( )
         {
            document.forms( "myForm" ).Strength.value =
               strengthIndex;
            document.forms( "myForm" ).Direction.value =
               ( blurDirection % 360 );

            if ( strengthIndex == 35 || strengthIndex == 0 )
               upDown = !upDown;

            blurImage.filters( "blur" ).strength =
               ( upDown ? strengthIndex++ : strengthIndex-- );

            if ( strengthIndex == 0 )
               blurImage.filters( "blur" ).direction =
                  ( ( blurDirection += 45 ) % 360 );
         }
         // -->
      </script>
   </head>

   <body>
      <form name = "myForm" action = "">

      <table border = "1" style = "background-color: #CCFFCC">
      <caption>Blur filter controls</caption>

         <tr>
            <td>Direction:</td>
            <td><select name = "Direction">
               <option value = "0">above</option>
               <option value = "45">above-right</option>
               <option value = "90">right</option>
               <option value = "135">below-right</option>
               <option value = "180">below</option>
               <option value = "225">below-left</option>
               <option value = "270">left</option>
               <option value = "315">above-left</option>
            </select></td>
         </tr>

         <tr>
            <td>Strength:</td>
            <td><input name = "Strength" size = "3" type = "text"
               maxlength = "3" value = "0" /></td>
         </tr>

         <tr>
            <td>Add original?</td>
            <td><input type = "checkbox" name = "AddBox" /></td>
         </tr>

         <tr>
            <td align = "center" colspan = "2">
               <input type = "button" value = "Apply"
                  onclick = "reBlur();" /></td>
         </tr>

         <tr>
            <td colspan = "2">
            <input type = "button" value = "Start demo"
              onclick = "startDemo();" />
            <input type = "button" value = "Stop demo"
              onclick = "window.clearInterval( timer );" /></td>
         </tr>

      </table>
       </form>

      <div id = "blurImage" style = "position: absolute;
         top: 0; left: 300; padding: 0; filter: blur(
         add = 0, direction = 0, strength = 0 );
         background-color: white;">
         <img align = "middle" src = "shapes.gif"
            alt = "Shapes" />
      </div>

   </body>
</html>
```

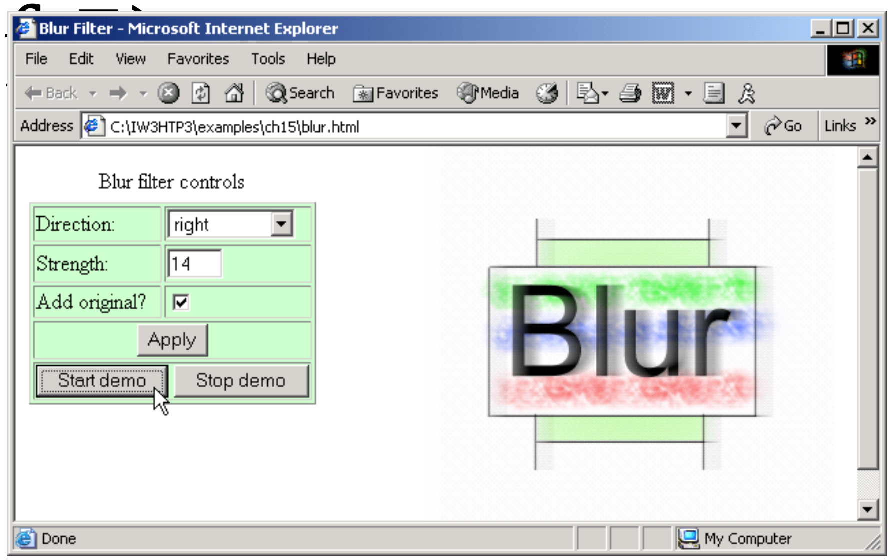
blur.html


Blur Filter - Microsoft Internet Explorer

Address: C:\IW3HTP3\examples\ch15\blur.html

Blur filter controls

Direction:	above-right
Strength:	22
Add original?	☐
	Apply
Start demo	Stop demo


Blur Filter - Microsoft Internet Explorer

Address: C:\IW3HTP3\examples\ch15\blur.html

Blur filter controls

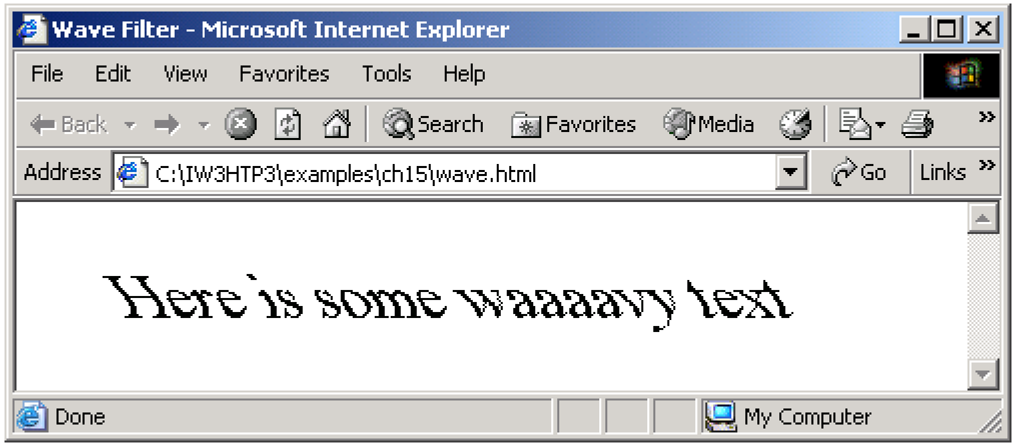
Direction:	right
Strength:	14
Add original?	☑
	Apply
Start demo	Stop demo

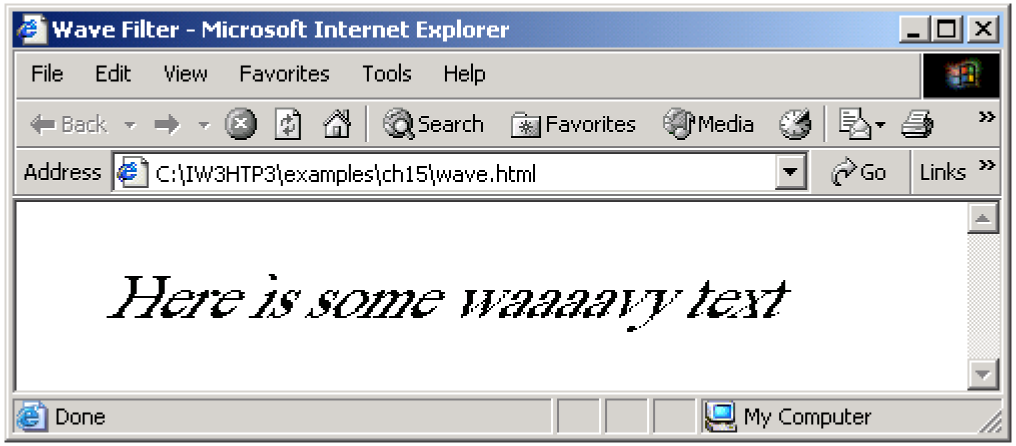
Using the `wave` Filter

- `wave` filter allows user to apply sine-wave distortions to text and images on Web pages
  - `add`
    - Adds a copy of the text or image underneath the filtered effect
  - `freq`
    - Determines the frequency of the wave applied
  - `phase`
    - Indicates the phase shift of the wave

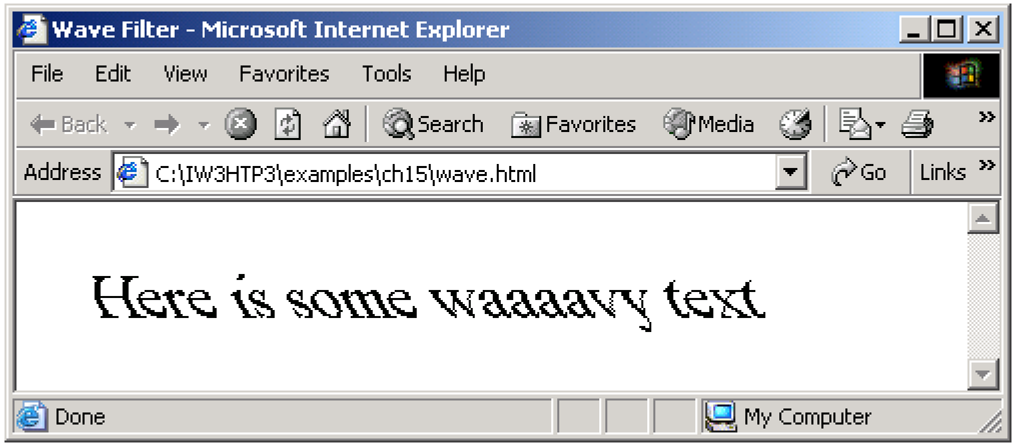
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.9: wave.html      -->
<!-- Applying the wave filter -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Wave Filter</title>

      <script type = "text/javascript">
         <!--
         var wavePhase = 0;

         function start()
         {
            window.setInterval( "wave()", 5 );
         }

         function wave()
         {
            wavePhase++;
            flag.filters( "wave" ).phase = wavePhase;
         }
```

```
        // -->
      </script>
   </head>

   <body onload = "start();">

      <span id = "flag"
         style = "align: center; position: absolute;
         left: 30; padding: 15;
         filter: wave(add = 0, freq = 1, phase = 0,
            strength = 10); font-size: 2em">
      Here is some waaaavy text
      </span>

   </body>
</html>
```


Wave Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\wave.html
Go Links
Here is some waaaavy text
Done
My Computer


Wave Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\wave.html Go Links

Here is some waaaavy text

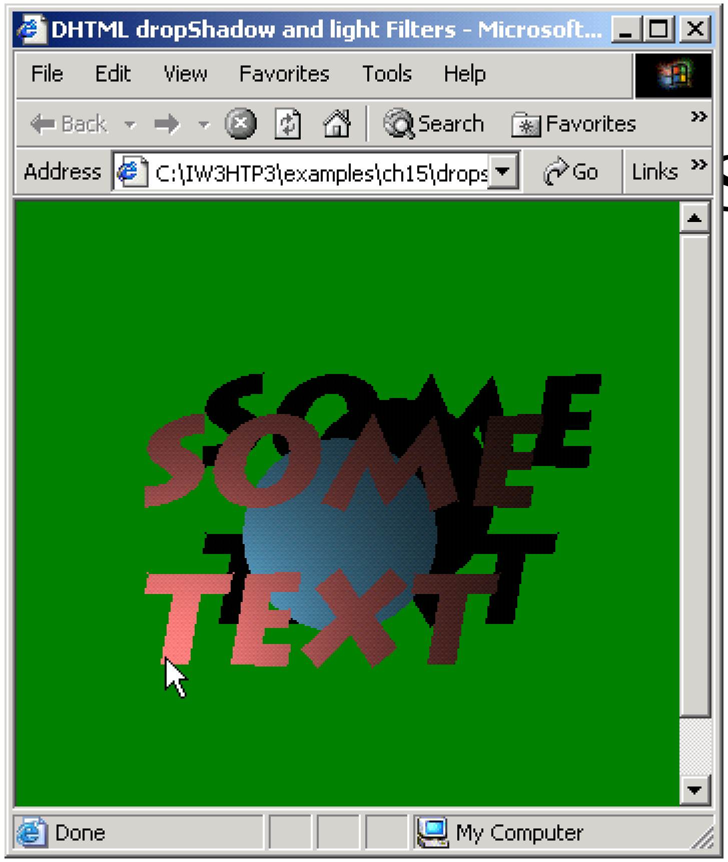

Wave Filter - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\wave.html Go Links

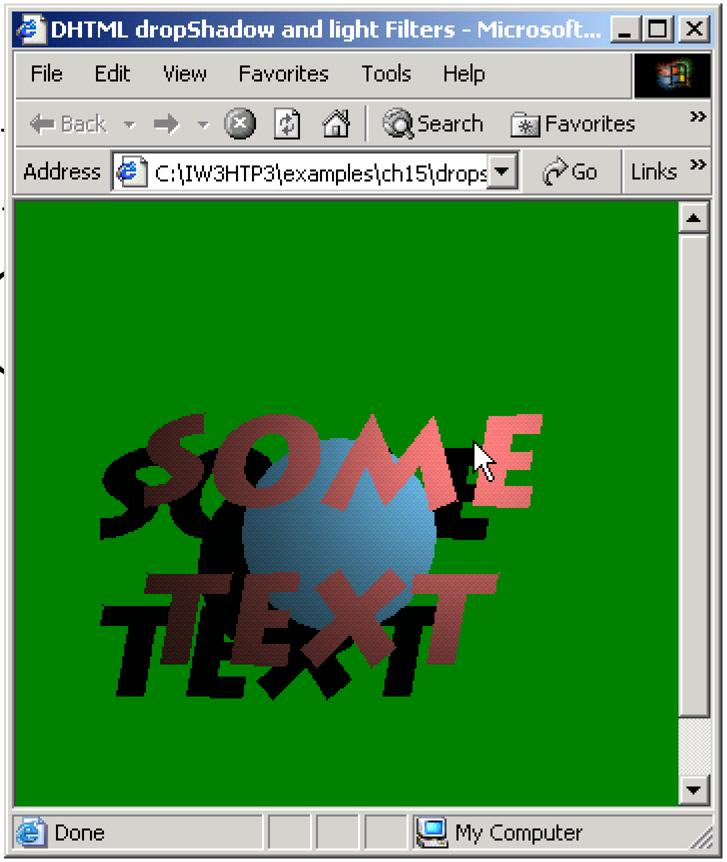
Here is some waaaavy text

Advanced Filters: `dropShadow` and `light`

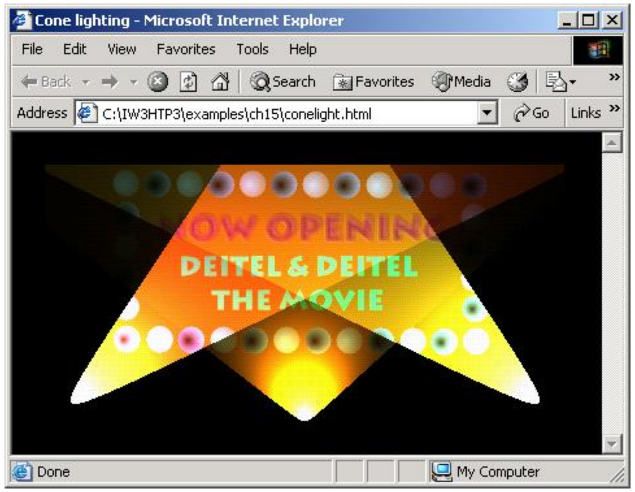
- `dropShadow`
  - Creates a blacked-out version of the image, and places it behind the image
  - `offx` and `offy` properties
    - Determined by how many pixels the drop shadow is offset
  - `color` property
    - Specifies the color of the drop shadow
- `light` filters
  - Most powerful and advanced filter in Internet Explorer 6.0
  - Allows simulation of a light source shining on Web page
  - All parameters and methods are set by scripting
  - `addPoint`
    - Adds a point light source

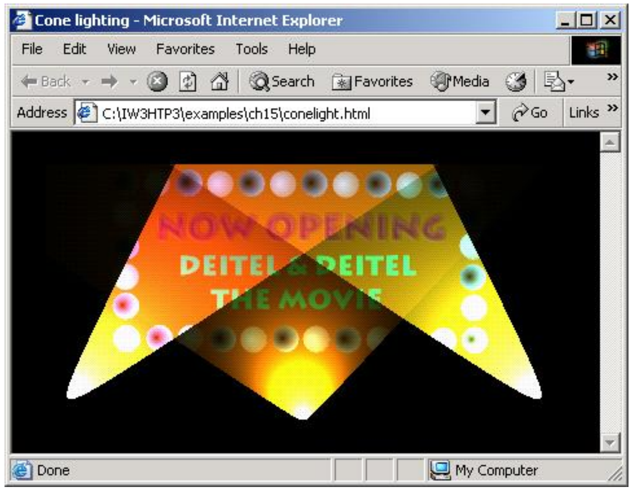
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 15.10: dropshadow.html                          -->
<!-- Using the light filter with the dropshadow filter   -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>DHTML dropShadow and light Filters</title>

      <script type = "text/javascript">
         <!--
         function setlight( )
         {
            dsImg.filters( "light" ).addPoint( 150, 150,
               125, 255, 255, 255, 100 );
         }

         function run()
         {
            eX = event.offsetX;
            eY = event.offsetY;

            xCoordinate = Math.round(
               eX-event.srcElement.width / 2, 0 );
            yCoordinate = Math.round(
               eY-event.srcElement.height / 2, 0 );

            dsImg.filters( "dropShadow" ).offx =
               xCoordinate / -3;
            dsImg.filters( "dropShadow" ).offy =
               yCoordinate / -3;

            dsImg.filters( "light" ).moveLight(
               0, eX, eY, 125, 1 );
         }
         // -->
      </script>
   </head>

   <body onload = "setlight()" style = "background-color: green">

      <img id = "dsImg" src = "circle.gif"
         style = "top: 100; left: 100; filter: dropShadow(
         offx = 0, offy = 0, color = black ) light()"
         onmousemove = "run()" alt = "Circle Image" />

   </body>
</html>
```


DHTML dropShadow and light Filters - Microsoft...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\drops
Go Links
SOME
TEXT
Done
My Computer


DHTML dropShadow and light Filters - Microsoft...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\drops
Go Links
SOME
TEXT
Done
My Computer

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.11: conelight.html        -->
<!-- Automating the cone light source -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head><title>Cone lighting</title>

    <script type = "text/javascript">
      var upDown = true;
      var counter = 0;
      var moveRate = -2;

      function setLight()
      {
         marquee.filters( "light" ).addCone( 0, marquee.height,
            8, marquee.width / 2, 30, 255, 150, 255, 50, 15 );
         marquee.filters( "light" ).addCone( marquee.width,
            marquee.height, 8, 200, 30, 150, 255, 255, 50, 15 );
         marquee.filters( "light" ).addCone( marquee.width / 2,
            marquee.height, 4, 200, 100, 255, 255, 150, 50, 50 );

         window.setInterval( "display()", 100 );
      }

      function display()
      {
         counter++;

         if ( ( counter % 30 ) == 0 )
            upDown = !upDown;

         if ( ( counter % 10 ) == 0 )
            moveRate *= -1;

         if ( upDown ) {
            marquee.filters( "light" ).moveLight(
               0, -1, -1, 3, 0 );
            marquee.filters( "light" ).moveLight(
               1, 1, -1, 3, 0 );
            marquee.filters( "light" ).moveLight(
               2, moveRate, 0, 3, 0);
         }
         else {
            marquee.filters( "light" ).moveLight(
               0, 1, 1, 3, 0 );
            marquee.filters( "light" ).moveLight(
               1, -1, 1, 3, 0 );
            marquee.filters( "light" ).moveLight(
               2, moveRate, 0, 3, 0) ;
         }
      }
   </script>
   </head>
   <body style = "background-color: #000000"
      onload = "setLight()">

      <img id = "marquee" src = "marquee.gif"
         style = "filter: light; position: absolute; left: 25;
         top: 25" alt = "Deitel movie marquee" />

   </body>
</html>
```


Cone lighting - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\conelight.html Go Links
NOW OPENING
DEITEL & DEITEL
THE MOVIE
Done
My Computer


Cone lighting - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address C:\IW3HTP3\examples\ch15\conelight.html Go Links
NOW OPENING
DEITEL & DEITEL
THE MOVIE
Done
My Computer

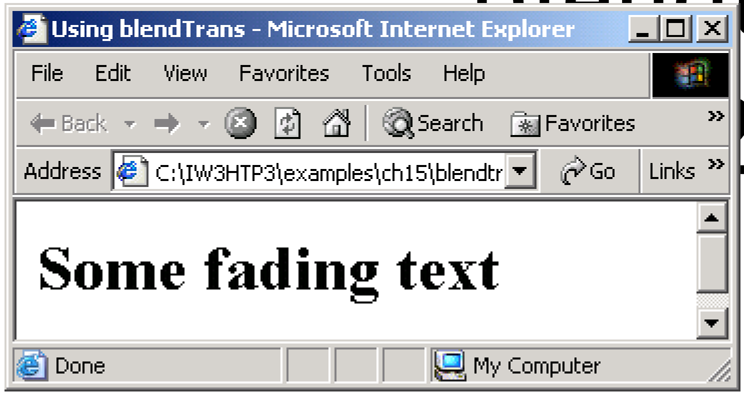
15.12 `blendTrans` Transition

- Example of the `blendTrans` transition
  - Creates a smooth fade-in/fade-out effect

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!-- Fig 15.12: blendtrans.html -->
<!-- Blend transition           -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Using blendTrans</title>

      <script type = "text/javascript">
         <!--
         function blendOut()
         {
            textInput.filters( "blendTrans" ).apply();
            textInput.style.visibility = "hidden";
            textInput.filters( "blendTrans" ).play();
         }
         // -->
      </script>
   </head>
   <body>

      <div id = "textInput" onclick = "blendOut()" style =
         "width: 300; filter: blendTrans( duration = 3 )">
         <h1>Some fading text</h1>
      </div>

   </body>
</html>
```

blendtrans.html

2 o

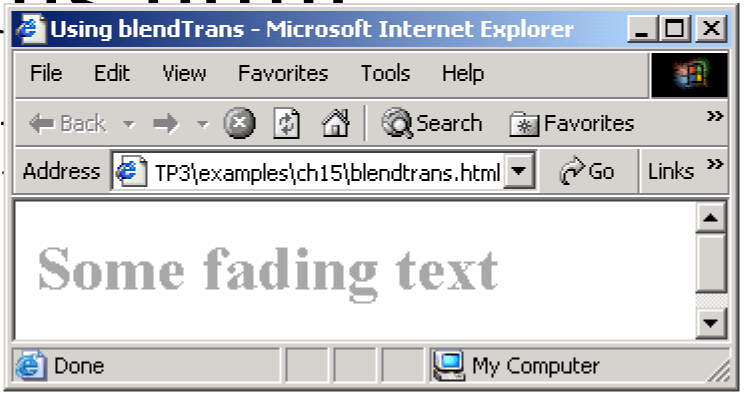

Using blendTrans - Microsoft Internet Explorer

Address C:\IW3HTP3\examples\ch15\blendtr

Some fading text

Done

My Computer


Using blendTrans - Microsoft Internet Explorer

Address TP3\examples\ch15\blendtrans.html

Some fading text

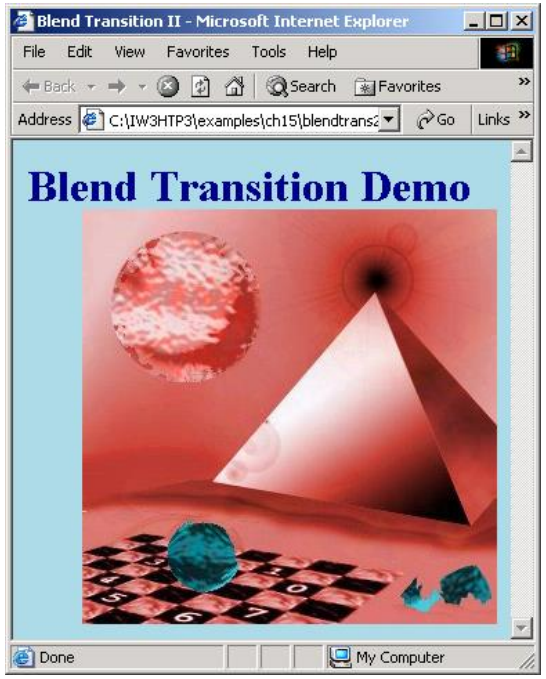
Done

My Computer

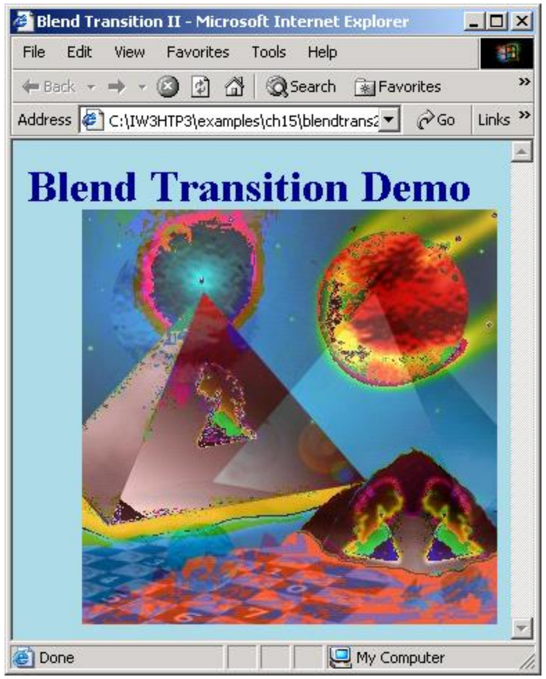
```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig 15.13: blendtrans2.html -->
<!-- Blend Transition            -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>Blend Transition II</title>

      <script type = "text/javascript">
         <!--
         var whichImage = true;

         function blend()
         {
            if ( whichImage ) {
               image1.filters( "blendTrans" ).apply();
               image1.style.visibility = "hidden";
               image1.filters( "blendTrans" ).play();
            }
            else {
               image2.filters( "blendTrans" ).apply();
               image2.style.visibility = "hidden";
```

```
                image2.filters( "blendTrans" ).play();
             }
          }

          function reBlend( fromImage )
          {
             if ( fromImage ) {
                image1.style.zIndex -= 2;
                image1.style.visibility = "visible";
             }
             else {
                image1.style.zIndex += 2;
                image2.style.visibility = "visible";
             }

             whichImage = !whichImage;
             blend();
          }
          // -->
       </script>
    </head>

    <body style = "color: darkblue; background-color: lightblue"
          onload = "blend()">

```

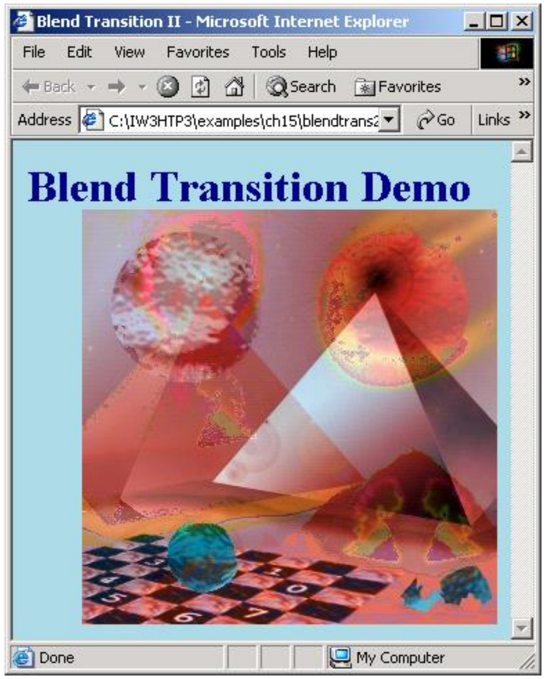
```
        <h1>Blend Transition Demo</h1>

        <img id = "image2" src = "cool12.jpg"
        onfilterchange = "reBlend( false )"
        style = "position: absolute; left: 50; top: 50;
        width: 300; filter: blendTrans( duration = 4 );
        z-index: 1" alt = "First Transition Image"  />

        <img id = "image1" src = "cool8.jpg"
           onfilterchange = "reBlend( true )"
           style = "position: absolute; left: 50; top: 50;
           width: 300; filter: blendTrans( duration = 4 );
           z-index: 2" alt = "Second Transition Image"  />

    </body>
</html>
```


Blend Transition II - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\blendtrans2
Go Links
Done
My Computer

Blend Transition Demo


Blend Transition II - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\blendtrans2
Go Links
Done
My Computer

Blend Transition Demo


Blend Transition II - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\blendtrans2
Go Links
Done
My Computer

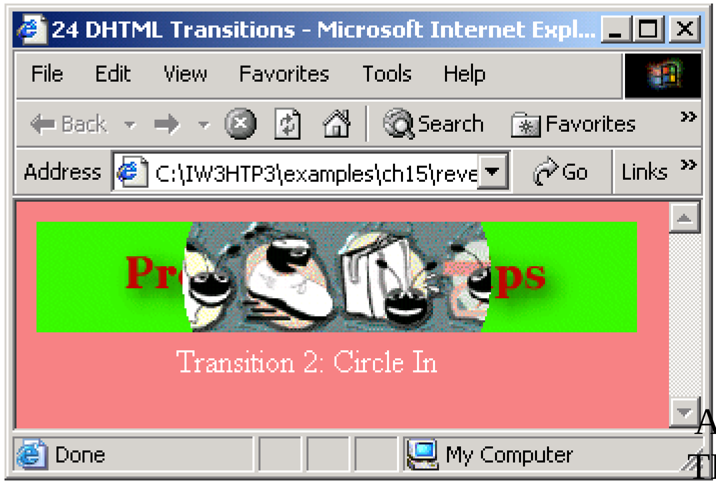
Blend Transition Demo

15.13 `revealTrans` Transition

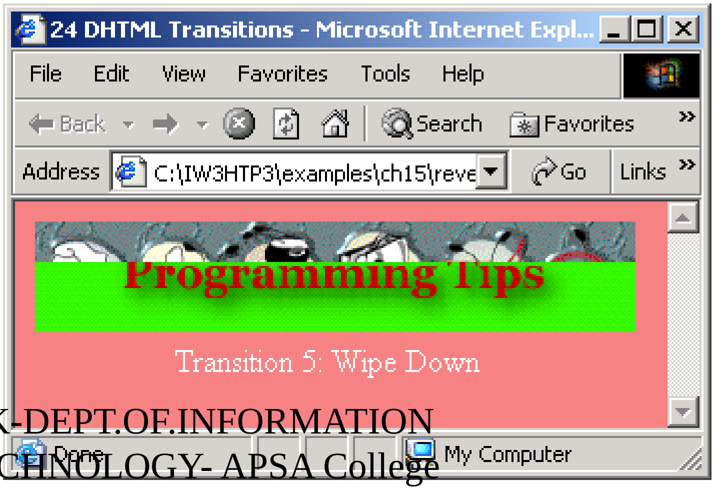
- `revealTrans` filter
  - Create professional-style transitions
  - From box out to random dissolve

```
<?xml version = "1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<!-- Fig. 15.14: revealtrans.html   -->
<!-- Cycling through 24 transitions -->

<html xmlns = "http://www.w3.org/1999/xhtml">
   <head>
      <title>24 DHTML Transitions</title>

    <script type = "text/javascript">
      <!--
      var transitionName =
         ["Box In", "Box Out",
          "Circle In", "Circle Out",
          "Wipe Up", "Wipe Down", "Wipe Right", "Wipe Left",
          "Vertical Blinds", "Horizontal Blinds",
          "Checkerboard Across", "Checkerboard Down",
          "Random Dissolve",
          "Split Vertical In", "Split Vertical Out",
          "Split Horizontal In", "Split Horizontal Out",
          "Strips Left Down", "Strips Left Up",
          "Strips Right Down", "Strips Right Up",
          "Random Bars Horizontal", "Random Bars Vertical",
          "Random" ];
       var counter = 0;
       var whichImage = true;

       function blend()
       {
          if ( whichImage ) {
             image1.filters( "revealTrans" ).apply();
             image1.style.visibility = "hidden";
             image1.filters( "revealTrans" ).play();
          }
          else {
             image2.filters( "revealTrans" ).apply();
             image2.style.visibility = "hidden";
             image2.filters( "revealTrans" ).play();
          }
       }

       function reBlend( fromImage )
       {
          counter++;

          if ( fromImage ) {
             image1.style.zIndex -= 2;
             image1.style.visibility = "visible";
```

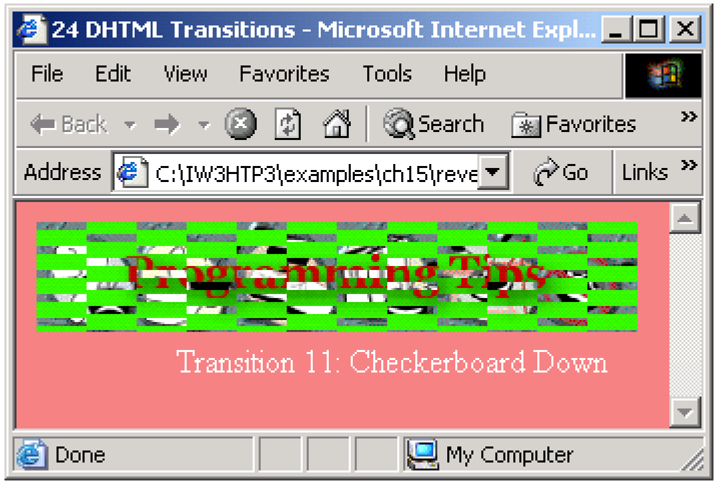
```
               image2.filters( "revealTrans" ).transition =
                  counter % 24;
            }
            else {
               image1.style.zIndex += 2;
               image2.style.visibility = "visible";
               image1.filters( "revealTrans" ).transition =
                  counter % 24;
            }

            whichImage = !whichImage;
            blend();
            transitionDisplay.innerHTML = "Transition " +
               counter % 24 + ": " + transitionName[ counter % 24 ];
         }
         // -->
      </script>
      </head>

      <body style = "color: white; background-color: lightcoral"
            onload = "blend()">

         <img id = "image2" src = "icontext.gif"
              style = "position: absolute; left: 10; top: 10;
              width: 300; z-index:1; visibility: visible;
              filter: revealTrans( duration = 2, transition = 0 )"
              onfilterchange = "reBlend( false )" alt =
              "Programming Tips" />

         <img id = "image1" src = "icons2.gif"
              style = "position: absolute; left: 10; top: 10;
              width: 300; z-index:1; visibility: visible;
              filter: revealTrans( duration = 2, transition = 0 )"
              onfilterchange = "reBlend( true )" alt = "Icons" />

         <div id = "transitionDisplay" style = "position: absolute;
            top: 70; left: 80">Transition 0: Box In</div>

      </body>
   </html>
```


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\reve
Go Links
Programming Tips
Done
My Computer

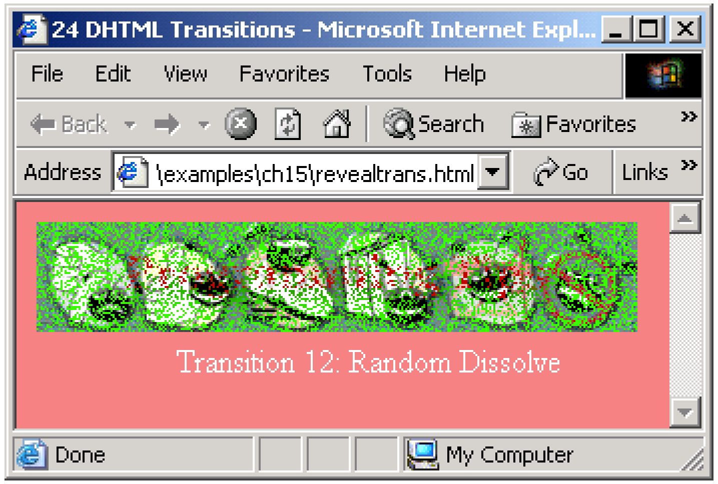
Transition 2: Circle In


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\reve
Go Links
Programming Tips
Done
My Computer

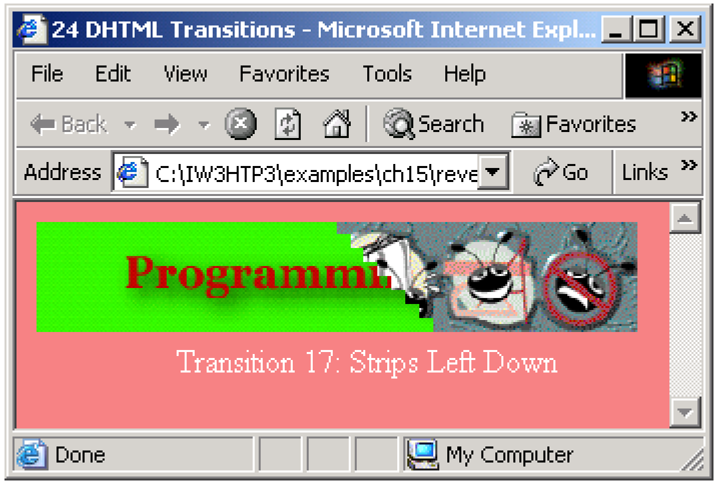
Transition 5: Wipe Down


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\reve
Go Links
Programming Tips
Done
My Computer

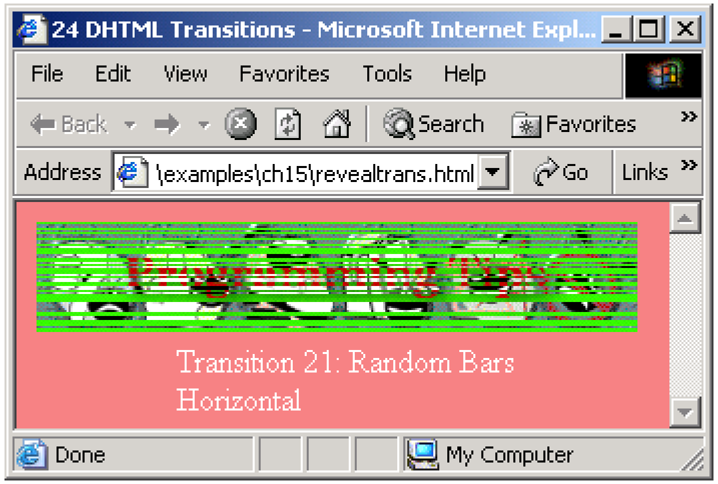
Transition 11: Checkerboard Down


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address \examples\ch15\revealtrans.html
Go Links
Done
My Computer

Transition 12: Random Dissolve


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address C:\IW3HTP3\examples\ch15\reve
Go Links
Programmi
Done
My Computer

Transition 17: Strips Left Down


24 DHTML Transitions - Microsoft Internet Expl...
File Edit View Favorites Tools Help
Back Search Favorites
Address \examples\ch15\revealtrans.html
Go Links
Programming Tips
Done
My Computer

Transition 21: Random Bars Horizontal