JavaScript and the HTML DOM

HTML Syntax

```
<tagname attribute="value" data-customattribute="value">
  More HTML tags or text
</tagname>
```

```
<tagname attribute="value" data-customattribute="value">
```

Example 1 – HTML says “Hello!”

```html
<!DOCTYPE html> <!-- specifies HTML5 standards mode -->
<html>
  <head>
    <!-- Page metadata (e.g. title, styles) lives here -->
  </head>
  <body>
    <!-- Display elements and document content goes here -->
    <h1>Hello, world!</h1> <!-- A top-level heading -->
  </body>
</html>
```

Including scripts in HTML

Example 2 – Scriptacular scripts!

```
<script type='application/javascript'>
  console.log('I am a script running directly in the page.');
</script>

<script type='application/javascript' src='external_script.js'>
//Anything here is ignored
</script>


/* external_script.js */
//Note that this file does not contain <script> tags
console.log("I am a script loaded from an external file!");
```

How to point to the scripts

Example 3 – Relative paths are absolutely amazing!

```
<!--
HTML loaded from http://introjsiap.com/examples/js/index.html
-->

<script type='application/javascript' src='../../myfile.js'>
//Points to http://introjsiap.com/myfile.js
</script>

<script type='application/javascript'
src='http://introjsiap.com/scripts/myfile.js'>
//Points to http://introjsiap.com/scripts/myfile.js
</script>
```

Where do the <script> tags go?

Example 4 – We want to know: where does the <script> go?

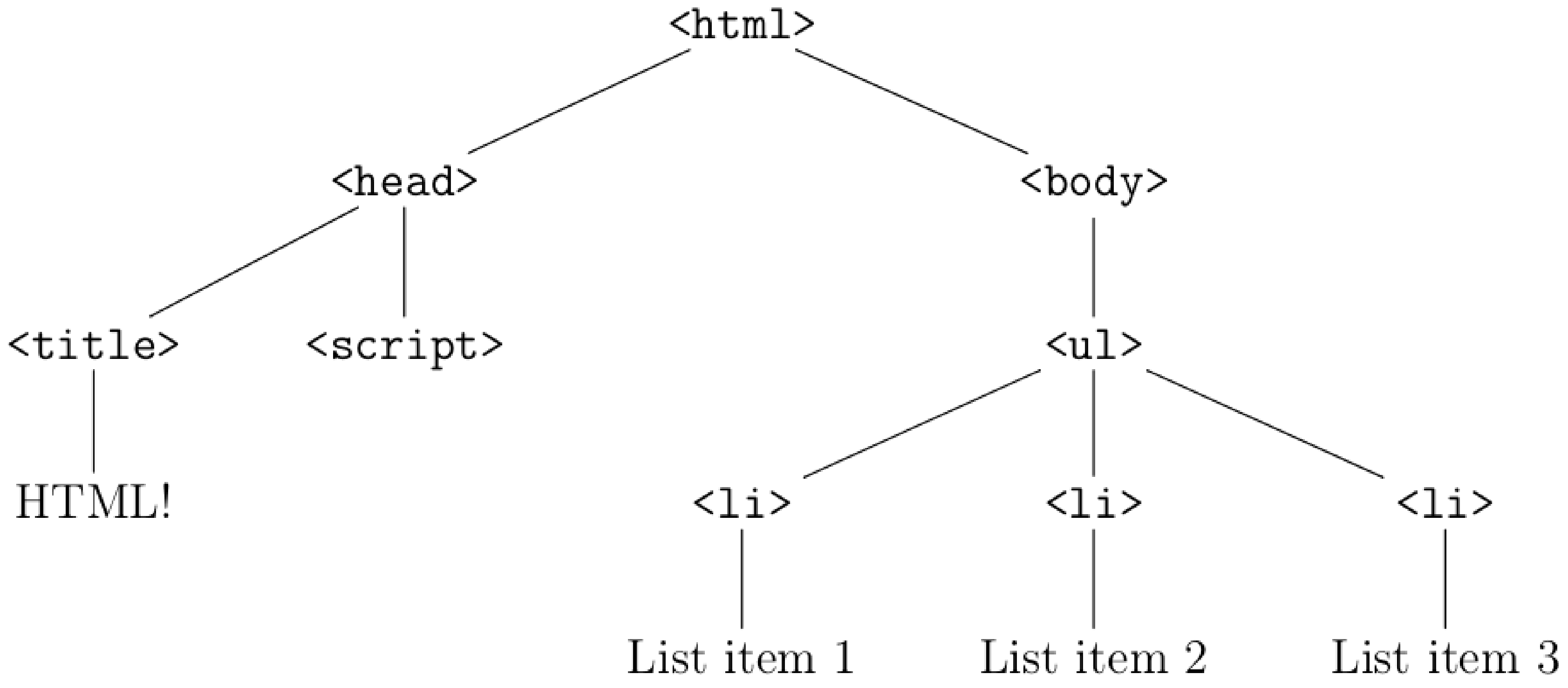
```
<!DOCTYPE html>
<html>
  <head>
    <!-- styles, metadata, title, etc. -->
    <!-- scripts that draw/render/create the page go here -->
    <script type='application/javascript'></script>
  </head>
  <body>
    <p>Other HTML elements and text</p>
    <!-- other scripts go here -->
    <script type='application/javascript'></script>
  </body>
</html>
```

The HTML DOM

Example 5 – In the domain of the DOM

```
<html>
  <head>
    <title>HTML!</title>
    <script type='application/javascript' src='draw_page.js'>
    </script>
  </head>
  <body>
    <ul> <!-- an unordered list -->
      <li>List item 1</li>
      <li>List item 2</li>
      <li>List item 3</li>
    </ul>
  </body>
</html>
```

The HTML DOM (in tree form)



An HTML template

Example 6 – The HTML page to begin all HTML pages

```
<!DOCTYPE html>
<html>
  <head>
    <title>The title of the page</title>
    <meta charset='utf-8'> <!-- the text encoding -->
  </head>
  <body>
  </body>
</html>
```

Example 7 – Meet the HTML family

```
<html>
<!-- <head> omitted for brevity -->
<body>
  <div> <!-- a page div[ision] -->
    <span id='someText'>This is some text.</span>
  </div>
  <script type='application/javascript' src='DOMRecursion.js'>
  </script>
</body>
</html>
```

Recursive traversal: recursal?

```
/* recursiveDOM.js */
//Functions similarly to document.getElementById()
function reimplementTheWheelById(id, parentNode) {
    //If parentNode is undefined, set it to the root node
    parentNode = parentNode || document.documentElement;

    if(parentNode.id === id) {
        return parentNode;
    }

    var children = parentNode.children;
    for(var i=0; i < children.length; i++) {
        var found = reimplementTheWheelById(id, children[i]);
        if(found !== null) {
            return found;
        }
    }
    return null;
}

var someText = reimplementTheWheelById("someText");
console.log(someText.innerHTML); //"This is some text."
```

Moving up and down the tree

```
DOMNode.children; //returns an HTMLCollection of child nodes
                  //HTMLCollection is just a fancy Array

$jqObj.children(); //returns the children as a jQuery object
                   //jQuery objects are really fancy Arrays

//Also,
DOMNode.parentNode; //returns the parent DOM node

$jqObj.parent();    //returns the parent jQuery object
```

Selecting elements

By ID:

```
/* DOM method */
document.getElementById("theID"); //returns a DOM node

/* jQuery method */
$("#theID"); //returns a jQuery object wrapping a DOM node
```

By tag name:

```
/* DOM method */
document.getElementsByTagName("tagname");
element.getElementsByTagName("tagname");
//returns an HTMLCollection (Array) of <tagname> elements

/* jQuery method */
$("tagname"); //returns the same elements as a jQuery object
```

By class:

```
/* DOM method */
document.getElementsByClassName("class1 optionalClass2");
element.getElementsByClassName("class1 optionalClass2");

/* jQuery method */
$(".class1 .optionalClass2");
```

Selecting by class

Example 8 – Keeping it classy

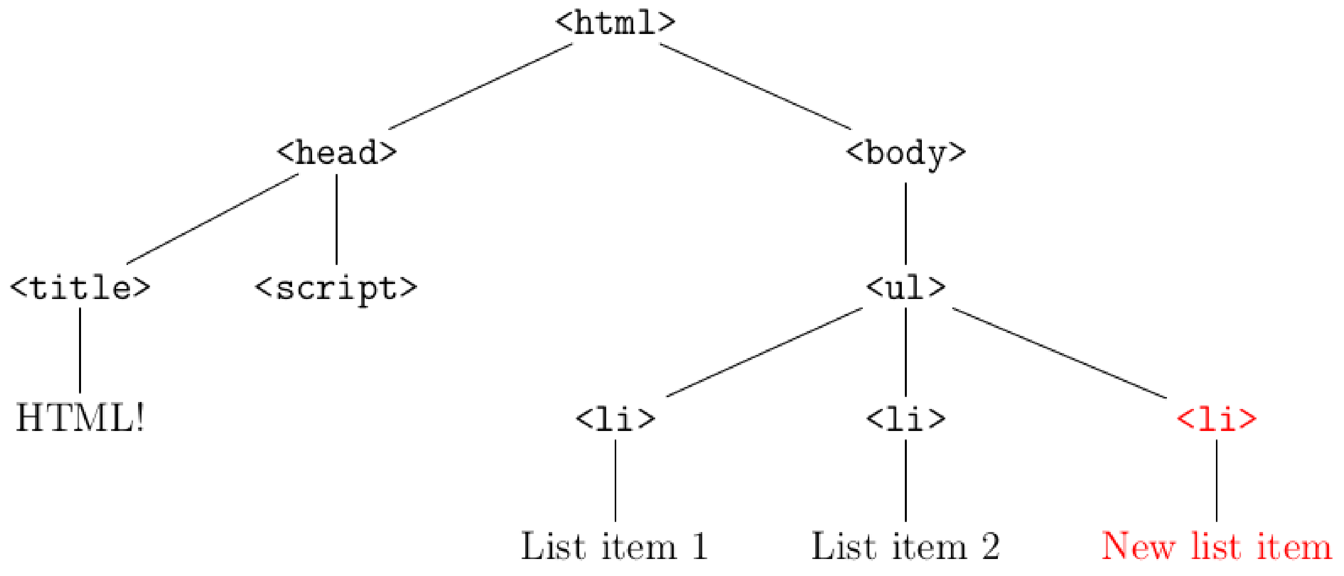
```
<html>
  <body>
    <p class='bodyText flavorText'>
      <span class='fourthWall'>
        Hmm. I seem to be in an example.
      </span>
    </p>
    <span class='bodyText'>Text, everywhere!</span>
  </body>
  <script type='application/javascript'
          src='jquery.min.js'></script>
  <script type='application/javascript'>
    //selects both the <p> and the second <span>
    var bodyText = document.getElementsByClassName("bodyText");
    //selects just the span
    var sr = bodyText[0].getElementsByClassName("fourthWall");

    var $bodyText = $(".bodyText");
    var $sr = $bodyText.find(".fourthWall");
  </script>
</html>
```

Selecting using CSS selectors

```
/* DOM method */
document.querySelector(selectors);

/* jQuery method */
$(selectors);

//selectors is a space-delimited String of CSS selectors
```

Adding HTML elements

```
/* DOM method */
var newElement = document.createElement("tagname");
otherElement.appendChild(newElement); //insert at the end
otherElement.insertBefore(newElement, beforeElement);

/* jQuery method */
var $newElement = $("<newElementTagName>");
$otherElem.append($newElement);
$newElement.appendTo(other);
//other can be a selector, jQuery object or DOM node
//$otherElem is a jQuery object
//$newElement is appended to every element in the selection
```

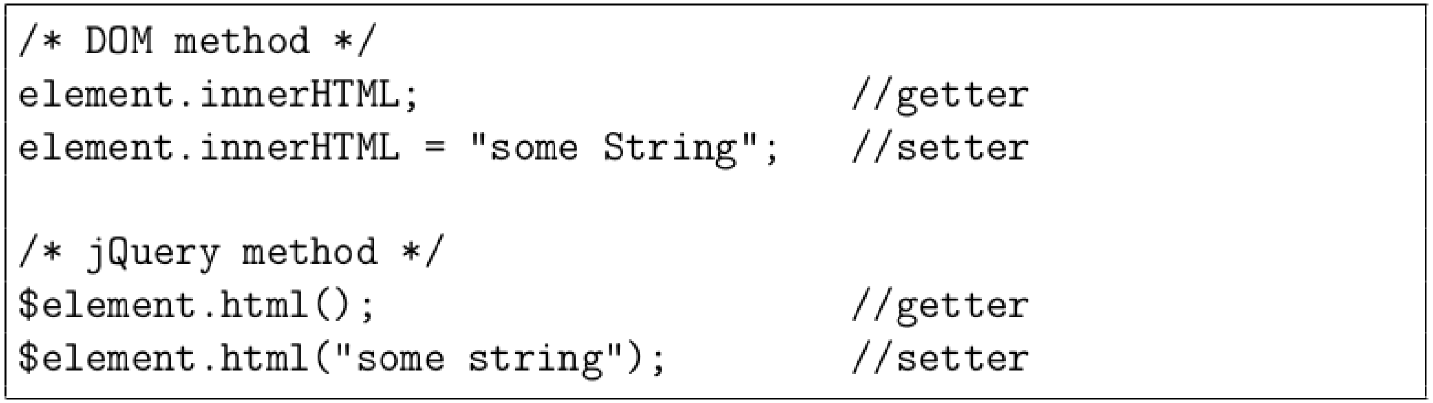


Removing elements

```
/* DOM method */
var removeMe = document.getElementById("removeMe");
var removed = removeMe.parentNode.removeChild(removeMe);
//returns the removed element

/* jQuery method */
var $removeThese = $("someSelector");
$removeThese.remove();
$removeThese.detach(); //removes but preserves jQuery data
```

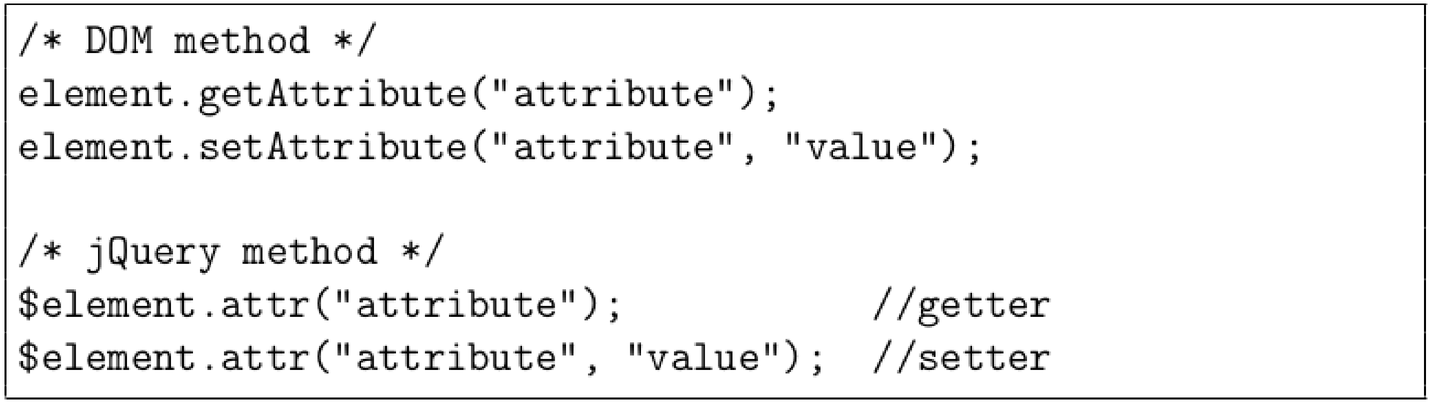
Modifying HTML elements

Inner HTML:



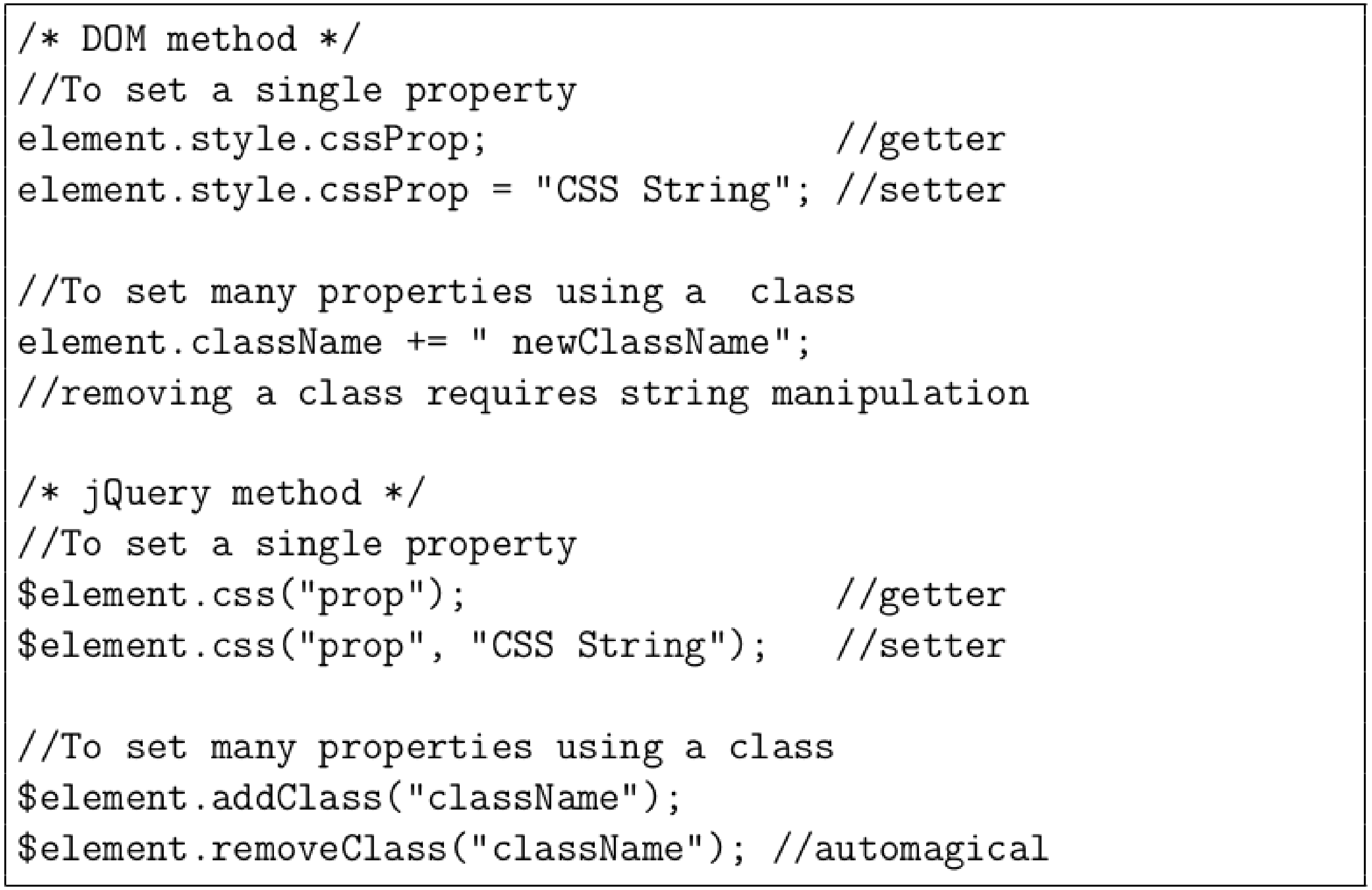
```
/* DOM method */
element.innerHTML;                          //getter
element.innerHTML = "some String";   //setter

/* jQuery method */
$element.html();                            //getter
$element.html("some string");         //setter
```

Attributes:



```
/* DOM method */
element.getAttribute("attribute");
element.setAttribute("attribute", "value");

/* jQuery method */
$element.attr("attribute");              //getter
$element.attr("attribute", "value");  //setter
```

Modifying elements' CSS



```
/* DOM method */
//To set a single property
element.style.cssProp;                //getter
element.style.cssProp = "CSS String"; //setter

//To set many properties using a  class
element.className += " newClassName";
//removing a class requires string manipulation

/* jQuery method */
//To set a single property
$element.css("prop");                 //getter
$element.css("prop", "CSS String");   //setter

//To set many properties using a class
$element.addClass("className");
$element.removeClass("className"); //automagical
```

CSS

Example 9 – Sí, es CSS

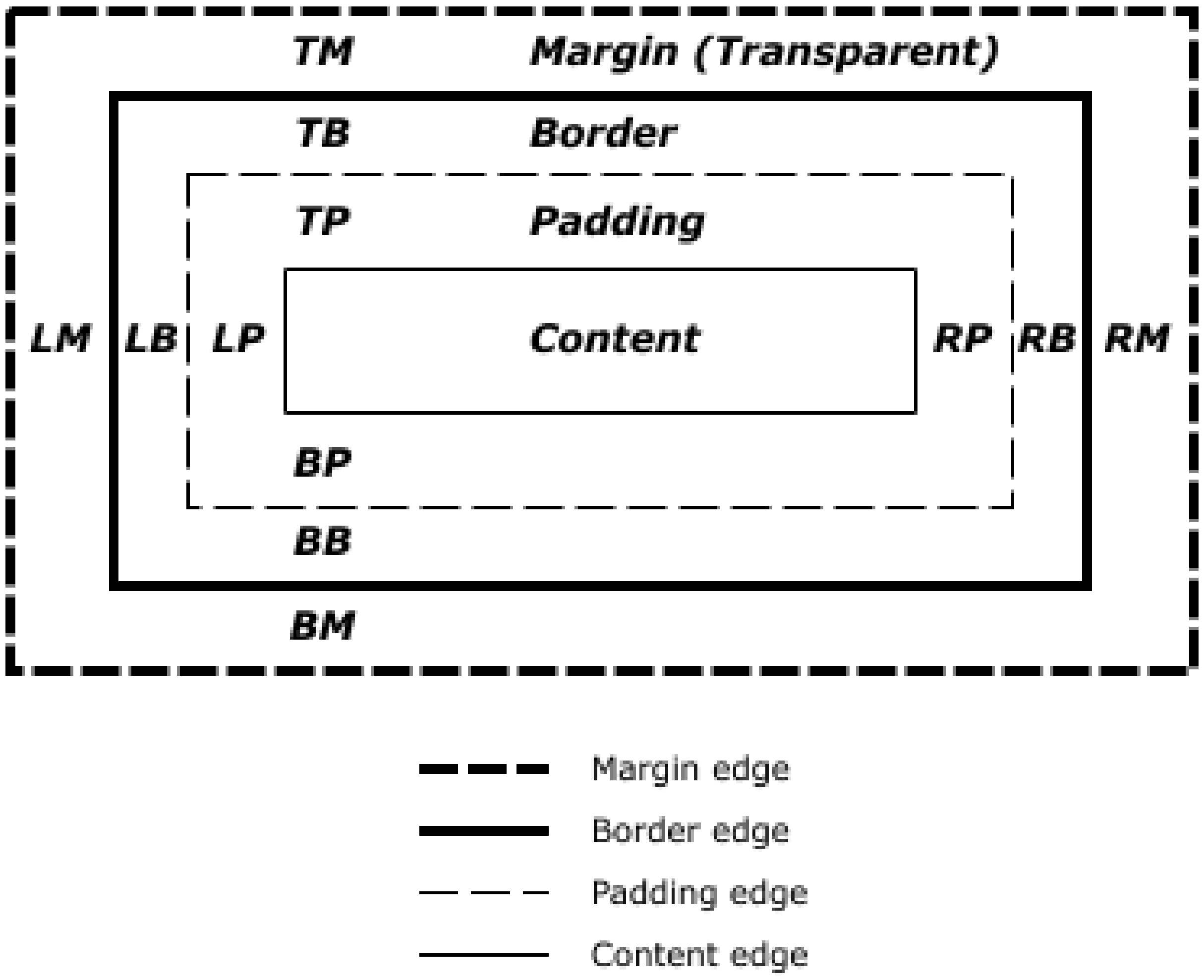
```html
<html>
<head>
  <style type='text/css'>
    div { /* selects all divs */
      width: 100px;
      height: 100px;
      background-color: #00ff00;
    }
    #divToo { background-color: #ff0000; }
  </style>
</head>
<body>
  <div>
    I will look like a green square.
  </div>
  <div id='divToo'>
    I will look like a red square.
  </div>
</body>
</html>
```

CSS syntax and where to put it

```
selector { /* This is a comment */
  property-name: value;
}
/* in JavaScript, property-name becomes propertyName */
```

```
<style type='text/css'>
  /* styles go here */
</style>

<link rel='stylesheet' type='text/css'
      href='path/to/stylesheet.css'>
<!-- basically the same as including JS -->
```

The CSS Box Model


TM
Margin (Transparent)
TB
Border
TP
Padding
LM
LB
LP
Content
RP
RB
RM
BP
BB
BM
Margin edge
Border edge
Padding edge
Content edge