18. JavaScript HTML DOM

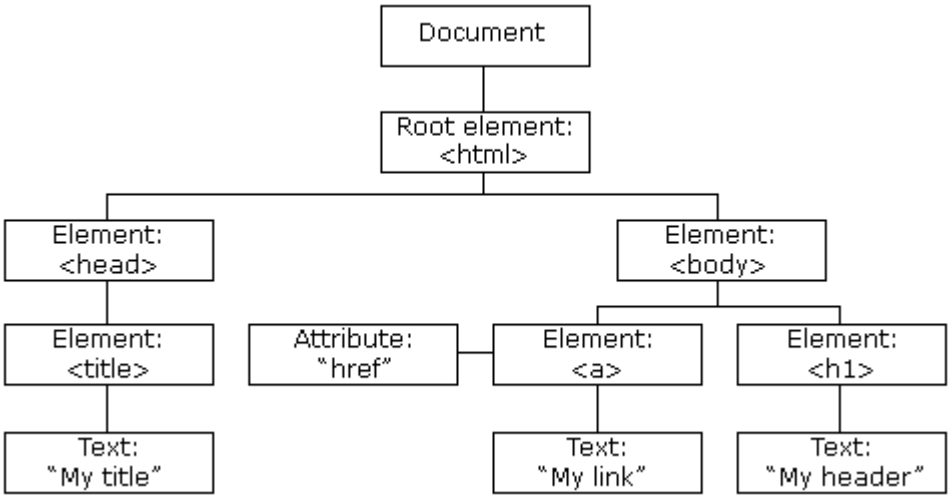
With the HTML DOM, JavaScript can access all the elements of an HTML document.

18.1. The HTML DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree



With a programmable object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can react to all the events in the page

18.2. Finding HTML Elements

Often, with JavaScript, you want to manipulate HTML elements.

To do so, you have to find the elements first. There are a couple of ways to do this:

- Finding HTML elements by id
- Finding HTML elements by tag name
- Finding HTML elements by class name

a) Finding HTML Elements by Id

The easiest way to find HTML elements in the DOM, is by using the element id.

This example finds the element with id="intro":

Example

```
var x=document.getElementById("intro");
```

If the element is found, the method will return the element as an object (in x).

If the element is not found, x will contain null.

b) Finding HTML Elements by Tag Name

This example finds the element with id="main", and then finds all <p> elements inside "main":

Example

```
var x=document.getElementById("main");
var y=x.getElementsByTagName("p");
```

Finding elements by class name does not work in Internet Explorer 5,6,7, and 8.

18.3. HTML DOM Tutorial

In the next pages of this tutorial you will learn:

- How to change the content (innerHTML) of HTML elements
- How to change the style (CSS) of HTML elements
- How to react to HTML DOM events
- How to add or delete HTML elements

I) JavaScript HTML DOM - Changing HTML

The HTML DOM allows JavaScript to change the content of HTML elements.

a) Changing the HTML Output Stream

JavaScript can create dynamic HTML content:

```
Date: Fri Jun 28 2013 13:25:15 GMT+0200 (CEST)
```

In JavaScript, document.write() can be used to write directly to the HTML output stream:

Example

```
<!DOCTYPE html>
<html>
<body>

<script>
document.write(Date());
</script>

</body>
</html>
```

Never use document.write() after the document is loaded. It will overwrite the document.

b) Changing HTML Content

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use this syntax:

```
document.getElementById(id).innerHTML=new HTML
```

This example changes the content of a <p> element:

Example

```
<html>
```

```
<body>

<p id="p1">Hello World!</p>

<script>
document.getElementById("p1").innerHTML="New
text!";
</script>

</body>
</html>
```

This example changes the content of an <h1> element:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1 id="header">Old Header</h1>

<script>
var element=document.getElementById("header");
element.innerHTML="New Header";
</script>

</body>
</html>
```

Example explained:

- The HTML document above contains an <h1> element with id="header"
- We use the HTML DOM to get the element with id="header"
- A JavaScript changes the content (innerHTML) of that element

c) Changing an HTML Attribute

To change the attribute of an HTML element, use this syntax:

```
document.getElementById(id).attribute=new value
```

This example changes the src attribute of an <img> element:

Example

```
<!DOCTYPE html>
<html>
<body>

<img id="image" src="smiley.gif">

<script>
document.getElementById("image").src="landscape.jpg
";
</script>

</body>
</html>
```

Example explained:

- The HTML document above contains an <img> element with id="image"
- We use the HTML DOM to get the element with id="image"
- A JavaScript changes the src attribute of that element from "smiley.gif" to "landscape.jpg"

II) JavaScript HTML DOM - Changing CSS

The HTML DOM allows JavaScript to change the style of HTML elements.

a) Changing HTML Style

To change the style of an HTML element, use this syntax:

```
document.getElementById(id).style.property=new style
```

The following example changes the style of a <p> element:

Example

```
<html>
<body>

<p id="p2">Hello World!</p>

<script>
document.getElementById("p2").style.color="blue";
</script>

<p>The paragraph above was changed by a script.</p>

</body>
</html>
```

This example changes the style of the HTML element with id="id1", when the user clicks a button:

Example

```
<!DOCTYPE html>
<html>
<body>

<h1 id="id1">My Heading 1</h1>
<button type="button"
onclick="document.getElementById('id1').style.color
='red'">
Click Me!</button>
```

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

b) More Examples

Visibility: How to make an element invisible. Do you want the element to show or not?

```
    <!DOCTYPE html>
    <html>
    <body>

    <p id="p1">This is a text. This is a text. This is
    a text. This is a text. This is a text. This is a
    text. This is a text.</p>

    <input type="button" value="Hide text"
    onclick="document.getElementById('p1').style.visibi
    lity='hidden'" />
    <input type="button" value="Show text"
    onclick="document.getElementById('p1').style.visibi
    lity='visible'" />

    </body>
    </html>
```

HTML DOM Style Object Reference

For all HTML DOM style properties, look at our complete HTML DOM Style Object Reference.

The Style object represents an individual style statement.

The Style object can be accessed from the document or from the elements to which that style is applied.

Syntax for using the Style object properties:

```
document.getElementById("id").style.property="value"
```

The Style object property categories:

- Background
- Border/Outline
- Generated Content
- List
- Misc
- Margin/Padding
- Positioning/Layout
- Printing
- Table
- Text

Background properties

W3C: W3C Standard.

Property	Description	W3C
background	Sets or returns all the background properties in one declaration	Yes
backgroundAttachment	Sets or returns whether a background-image is fixed or scrolls with the page	Yes
backgroundColor	Sets or returns the background-color of an element	Yes
backgroundImage	Sets or returns the background-image for an element	Yes
backgroundPosition	Sets or returns the starting position of a background-image	Yes
backgroundRepeat	Sets or returns how to repeat (tile) a background-image	Yes

Border/Outline properties

Property	Description	W3C
border	Sets or returns border-width, border-style, and border-color in one declaration	Yes
borderBottom	Sets or returns all the borderBottom* properties in one declaration	Yes
borderBottomColor	Sets or returns the color of the bottom border	Yes
borderBottomStyle	Sets or returns the style of the bottom border	Yes
borderBottomWidth	Sets or returns the width of the bottom border	Yes
borderColor	Sets or returns the color of an element's border (can have up to four values)	Yes
borderLeft	Sets or returns all the borderLeft* properties in one declaration	Yes
borderLeftColor	Sets or returns the color of the left border	Yes
borderLeftStyle	Sets or returns the style of the left border	Yes
borderLeftWidth	Sets or returns the width of the left border	Yes
borderRight	Sets or returns all the borderRight* properties in one declaration	Yes

borderRightColor	Sets or returns the color of the right border	Yes
borderRightStyle	Sets or returns the style of the right border	Yes
borderRightWidth	Sets or returns the width of the right border	Yes
borderStyle	Sets or returns the style of an element's border (can have up to four values)	Yes
borderTop	Sets or returns all the borderTop* properties in one declaration	Yes
borderTopColor	Sets or returns the color of the top border	Yes
borderTopStyle	Sets or returns the style of the top border	Yes
borderTopWidth	Sets or returns the width of the top border	Yes
borderWidth	Sets or returns the width of an element's border (can have up to four values)	Yes
outline	Sets or returns all the outline properties in one declaration	Yes
outlineColor	Sets or returns the color of the outline around a element	Yes
outlineStyle	Sets or returns the style of the outline around an element	Yes
outlineWidth	Sets or returns the width of the outline around an element	Yes

Generated Content Properties

Property	Description	W3C
content	Sets or returns the generated content before or after the element	Yes
counterIncrement	Sets or returns the list of counters and increment values	Yes
counterReset	Sets or returns the list of counters and their initial values	Yes

List properties

Property	Description	W3C
listStyle	Sets or returns list-style-image, list-style-position, and list-style-type in one declaration	Yes
listStyleImage	Sets or returns an image as the list-item marker	Yes

listStylePosition	Sets or returns the position of the list-item marker	Yes
listStyleType	Sets or returns the list-item marker type	Yes

Margin/Padding properties

Property	Description	W3C
margin	Sets or returns the margins of an element (can have up to four values)	Yes
marginBottom	Sets or returns the bottom margin of an element	Yes
marginLeft	Sets or returns the left margin of an element	Yes
marginRight	Sets or returns the right margin of an element	Yes
marginTop	Sets or returns the top margin of an element	Yes
padding	Sets or returns the padding of an element (can have up to four values)	Yes
paddingBottom	Sets or returns the bottom padding of an element	Yes
paddingLeft	Sets or returns the left padding of an element	Yes
paddingRight	Sets or returns the right padding of an element	Yes
paddingTop	Sets or returns the top padding of an element	Yes

Misc properties

Property	Description	W3C
cssText	Sets or returns the contents of a style declaration as a string	Yes

Positioning/Layout properties

Property	Description	W3C
bottom	Sets or returns the bottom position of a positioned element	Yes

clear	Sets or returns the position of the element relative to floating objects	Yes
clip	Sets or returns which part of a positioned element is visible	Yes
cssFloat	Sets or returns the horizontal alignment of an object	Yes
cursor	Sets or returns the type of cursor to display for the mouse pointer	Yes
display	Sets or returns an element's display type	Yes
height	Sets or returns the height of an element	Yes
left	Sets or returns the left position of a positioned element	Yes
maxHeight	Sets or returns the maximum height of an element	Yes
maxWidth	Sets or returns the maximum width of an element	Yes
minHeight	Sets or returns the minimum height of an element	Yes
minWidth	Sets or returns the minimum width of an element	Yes
overflow	Sets or returns what to do with content that renders outside the element box	Yes
position	Sets or returns the type of positioning method used for an element (static, relative, absolute or fixed)	Yes
right	Sets or returns the right position of a positioned element	Yes
top	Sets or returns the top position of a positioned element	Yes
verticalAlign	Sets or returns the vertical alignment of the content in an element	Yes
visibility	Sets or returns whether an element should be visible	Yes
width	Sets or returns the width of an element	Yes
zIndex	Sets or returns the stack order of a positioned element	Yes

Printing properties

Property	Description	W3C
orphans	Sets or returns the minimum number of lines for an element	Yes

	that must be visible at the bottom of a page	
pageBreakAfter	Sets or returns the page-break behavior after an element	Yes
pageBreakBefore	Sets or returns the page-break behavior before an element	Yes
pageBreakInside	Sets or returns the page-break behavior inside an element	Yes
widows	Sets or returns the minimum number of lines for an element that must be visible at the top of a page	Yes

Table properties

Property	Description	W3C
borderCollapse	Sets or returns whether the table border should be collapsed into a single border, or not	Yes
borderSpacing	Sets or returns the space between cells in a table	Yes
captionSide	Sets or returns the position of the table caption	Yes
emptyCells	Sets or returns whether to show the border and background of empty cells, or not	Yes
tableLayout	Sets or returns the way to lay out table cells, rows, and columns	Yes

Text properties

Property	Description	W3C
color	Sets or returns the color of the text	Yes
direction	Sets or returns the text direction	Yes
font	Sets or returns font-style, font-variant, font-weight, font-size, line-height, and font-family in one declaration	Yes
fontFamily	Sets or returns the font face for text	Yes
fontSize	Sets or returns the font size of the text	Yes
fontSizeAdjust	Sets or returns the font aspect value	Yes

fontStyle	Sets or returns whether the style of the font is normal, italic or oblique	Yes
fontVariant	Sets or returns whether the font should be displayed in small capital letters	Yes
fontWeight	Sets or returns the boldness of the font	Yes
letterSpacing	Sets or returns the space between characters in a text	Yes
lineHeight	Sets or returns the distance between lines in a text	Yes
quotes	Sets or returns the type of quotation marks for embedded quotations	Yes
textAlign	Sets or returns the horizontal alignment of text	Yes
textDecoration	Sets or returns the decoration of a text	Yes
textIndent	Sets or returns the indentation of the first line of text	Yes
textShadow	Sets or returns the shadow effect of a text	Yes
textTransform	Sets or returns the case of a text	Yes
unicodeBidi	Sets or returns whether the text should be overridden to support multiple languages in the same document	Yes
whiteSpace	Sets or returns how to handle tabs, line breaks and whitespace in a text	Yes
wordSpacing	Sets or returns the spacing between words in a text	Yes

III) JavaScript HTML DOM Events

HTML DOM allows JavaScript to react to HTML events.

Reacting to Events

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
onclick=JavaScript
```

Examples of HTML events:

- When a user clicks the mouse
- When a web page has loaded
- When an image has been loaded
- When the mouse moves over an element
- When an input field is changed
- When an HTML form is submitted
- When a user strokes a key

In this example, the content of the <h1> element is changed when a user clicks on it:

Example

```
<!DOCTYPE html>
<html>
<body>
<h1 onclick="this.innerHTML='Ooops!'">Click on this
text!</h1>
</body>
</html>
```

In this example, a function is called from the event handler:

Example

```
<!DOCTYPE html>
<html>
<head>
<script>
function changetext(id)
```

```
{
id.innerHTML="Ooops!";
}
</script>
</head>
<body>
<h1 onclick="changetext(this)">Click on this text!
</h1>
</body>
</html>
```

a) HTML Event Attributes

To assign events to HTML elements you can use event attributes.

Example

Assign an onclick event to a button element:

```
<button onclick="displayDate()">Try it</button>
```

In the example above, a function named *displayDate* will be executed when the button is clicked.

b) Assign Events Using the HTML DOM

The HTML DOM allows you to assign events to HTML elements using JavaScript:

Example

Assign an onclick event to a button element:

```
<script>
document.getElementById("myBtn").onclick=function()
{displayDate()};
</script>
```

In the example above, a function named *displayDate* is assigned to an HTML element with the id=myBtn".

The function will be executed when the button is clicked.

c) The onload and onunload Events

The onload and onunload events are triggered when the user enters or leaves the page.

The onload event can be used to check the visitor's browser type and browser version, and load the proper version of the web page based on the information.

The onload and onunload events can be used to deal with cookies.

Example

```
<body onload="checkCookies()">
```

d) The onchange Event

The onchange event are often used in combination with validation of input fields.

Below is an example of how to use the onchange. The upperCase() function will be called when a user changes the content of an input field.

Example

```
<input type="text" id="fname" onchange="upperCase()">
```

e) The onmouseover and onmouseout Events

The onmouseover and onmouseout events can be used to trigger a function when the user mouses over, or out of, an HTML element.

Example

A simple onmouseover-onmouseout example:

```
<!DOCTYPE html>
<html>
<body>

<div onmouseover="mOver(this)"
onmouseout="mOut(this)" style="background-
color:#D94A38;width:120px;height:20px;padding:40px;
">Mouse Over Me</div>
```

```
<script>
function mOver(obj)
{
obj.innerHTML="Thank You"
}

function mOut(obj)
{
obj.innerHTML="Mouse Over Me"
}
</script>

</body>
</html>
```

The onmousedown, onmouseup and onclick Events

The onmousedown, onmouseup, and onclick events are all parts of a mouse-click. First when a mouse-button is clicked, the onmousedown event is triggered, then, when the mouse-button is released, the onmouseup event is triggered, finally, when the mouse-click is completed, the onclick event is triggered.

Example

A simple onmousedown-onmouseup example:

```
<!DOCTYPE html>
<html>
<body>

<div onmousedown="mDown(this)"
onmouseup="mUp(this)" style="background-
color:#D94A38;width:90px;height:20px;padding:40px;"
>Click Me</div>

<script>
function mDown(obj)
{
obj.style.backgroundColor="#1ec5e5";
obj.innerHTML="Release Me"
}

function mUp(obj)
{
obj.style.backgroundColor="#D94A38";
obj.innerHTML="Thank You"
```

```
}
</script>

</body>
</html>
```

More Examples

a) onmousedown and onmouseup. Change an image when a user holds down the mouse button.

```
<!DOCTYPE html>
<html>
<head>
<script>
function lighton()
{
document.getElementById('myimage').src="bulbon.gif"
;
}
function lightoff()
{
document.getElementById('myimage').src="bulboff.gif
";
}
</script>
</head>

<body>
<img id="myimage" onmousedown="lighton()"
onmouseup="lightoff()" src="bulboff.gif"
width="100" height="180" />
<p>Click and hold to turn on the light!</p>
</body>
</html>
```

b) onload. Display an alert box when the page has finished loading.

```
<!DOCTYPE html>
<html>
<head>

<script>
function mymessage()
{
alert("This message was triggered from the onload
event");
```

```
}
</script>
</head>

<body onload="mymessage()">
</body>

</html>
```

c) onfocus. Change the background-color of an input field when it gets focus.

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction(x)
{
x.style.background="yellow";
}
</script>
</head>
<body>

Enter your name: <input type="text"
onfocus="myFunction(this)">

<p>When the input field gets focus, a function is
triggered which changes the background-color.</p>

</body>
</html>
```

d) Mouse Events. Change the color of an element when the cursor moves over it.

```
<!DOCTYPE html>
<html>
<body>

<h1 onmouseover="style.color='red'"
onmouseout="style.color='black'">
Mouse over this text</h1>

</body>
</html>
```

HTML DOM Event Object Reference

For a list of all HTML DOM events, look at our complete HTML DOM Event Object Reference.

HTML DOM events allow JavaScript to register different event handlers on elements in an HTML document. Events are normally used in combination with functions, and the function will not be executed before the event occurs (such as when a user clicks a button).

Tip: The event model was standardized by the W3C in DOM Level 2.

DOM: Indicates in which DOM Level the property was introduced.

Mouse Events

Property	Description	DOM
onclick	The event occurs when the user clicks on an element	2
ondblclick	The event occurs when the user double-clicks on an element	2
onmousedown	The event occurs when a user presses a mouse button over an element	2
onmousemove	The event occurs when the pointer is moving while it is over an element	2
onmouseover	The event occurs when the pointer is moved onto an element	2
onmouseout	The event occurs when a user moves the mouse pointer out of an element	2
onmouseup	The event occurs when a user releases a mouse button over an element	2

Keyboard Events

Attribute	Description	DOM
onkeydown	The event occurs when the user is pressing a key	2
onkeypress	The event occurs when the user presses a key	2
onkeyup	The event occurs when the user releases a key	2

Frame/Object Events

Attribute	Description	DOM
onabort	The event occurs when an image is stopped from loading before completely loaded (for <object>)	2
onerror	The event occurs when an image does not load properly (for <object>, <body> and <frameset>)	
onload	The event occurs when a document, frameset, or <object> has been loaded	2

onresize	The event occurs when a document view is resized	2
onscroll	The event occurs when a document view is scrolled	2
onunload	The event occurs once a page has unloaded (for <body> and <frameset>)	2

Form Events

Attribute	Description	DOM
onblur	The event occurs when a form element loses focus	2
onchange	The event occurs when the content of a form element, the selection, or the checked state have changed (for <input>, <select>, and <textarea>)	2
onfocus	The event occurs when an element gets focus (for <label>, <input>, <select>, textarea>, and <button>)	2
onreset	The event occurs when a form is reset	2
onselect	The event occurs when a user selects some text (for <input> and <textarea>)	2
onsubmit	The event occurs when a form is submitted	2

Event Object

Constants

Constant	Description	DOM
CAPTURING_PHASE	The current event phase is the capture phase (3)	1
AT_TARGET	The current event is in the target phase, i.e. it is being evaluated at the event target (1)	2
BUBBLING_PHASE	The current event phase is the bubbling phase (2)	3

Properties

Property	Description	DOM
bubbles	Returns whether or not an event is a bubbling event	2
cancelable	Returns whether or not an event can have its default action prevented	2
currentTarget	Returns the element whose event listeners triggered the event	2
eventPhase	Returns which phase of the event flow is currently being evaluated	2

target	Returns the element that triggered the event	2
timeStamp	Returns the time (in milliseconds relative to the epoch) at which the event was created	2
type	Returns the name of the event	2

Methods

Method	Description	DOM
initEvent()	Specifies the event type, whether or not the event can bubble, whether or not the event's default action can be prevented	2
preventDefault()	To cancel the event if it is cancelable, meaning that any default action normally taken by the implementation as a result of the event will not occur	2
stopPropagation()	To prevent further propagation of an event during event flow	2

EventTarget Object

Methods

Method	Description	DOM
addEventListener()	Allows the registration of event listeners on the event target (IE8 = attachEvent())	2
dispatchEvent()	Allows to send the event to the subscribed event listeners (IE8 = fireEvent())	2
removeEventListener()	Allows the removal of event listeners on the event target (IE8 = detachEvent())	2

EventListener Object

Methods

Method	Description	DOM
handleEvent()	Called whenever an event occurs of the event type for which the EventListener interface was registered	2

DocumentEvent Object

Methods

Method	Description	DOM

createEvent()		2

MouseEvent/KeyboardEvent Object

Properties

Property	Description	DOM
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	2
button	Returns which mouse button was clicked when an event was triggered	2
clientX	Returns the horizontal coordinate of the mouse pointer, relative to the current window, when an event was triggered	2
clientY	Returns the vertical coordinate of the mouse pointer, relative to the current window, when an event was triggered	2
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	2
keyIdentifier	Returns the identifier of a key	3
keyLocation	Returns the location of the key on the advice	3
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	2
relatedTarget	Returns the element related to the element that triggered the event	2
screenX	Returns the horizontal coordinate of the mouse pointer, relative to the screen, when an event was triggered	2
screenY	Returns the vertical coordinate of the mouse pointer, relative to the screen, when an event was triggered	2
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	2

Methods

Method	Description	W3C
initMouseEvent()	Initializes the value of a MouseEvent object	2
initKeyboardEvent()	Initializes the value of a KeyboardEvent object	3

IV) JavaScript HTML DOM Elements (Nodes)

Adding and Removing Nodes (HTML Elements)

a) Creating New HTML Elements

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

Example

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
var para=document.createElement("p");
var node=document.createTextNode("This is new.");
para.appendChild(node);

var element=document.getElementById("div1");
element.appendChild(para);
</script>
```

Example Explained

This code creates a new <p> element:

```
var para=document.createElement("p");
```

To add text to the <p> element, you must create a text node first. This code creates a text node:

```
var node=document.createTextNode("This is a new
paragraph.");
```

Then you must append the text node to the <p> element:

```
para.appendChild(node);
```

Finally you must append the new element to an existing element.

This code finds an existing element:

```
var element=document.getElementById("div1");
```

This code appends the new element to the existing element:

```
element.appendChild(para);
```

b) Removing Existing HTML Elements

To remove an HTML element, you must know the parent of the element:

Example

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
<script>
var parent=document.getElementById("div1");
var child=document.getElementById("p1");
parent.removeChild(child);
</script>
```

Example Explained

This HTML document contains a element with two child nodes (two <p> elements):

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>
```

Find the element with id="div1":

```
var parent=document.getElementById("div1");
```

Find the <p> element with id="p1":

```
var child=document.getElementById("p1");
```

Remove the child from the parent:

```
parent.removeChild(child);
```

It would be nice to be able to remove an element without referring to the parent.
But sorry. The DOM needs to know both the element you want to remove, and its parent.
Here is a common workaround: Find the child you want to remove, and use its parentNode property to find the parent:

```
var child=document.getElementById("p1");
child.parentNode.removeChild(child);
```