Module 4 Cheatsheet: JavaScript Programming for Web Applications

Class or Method	Description	Example

`appendChild()`	An HTML DOM method that after creating an element, you can use this function to place the element in the appropriate location within the document. The element to append is the only parameter.	`//Creates the element <p> and text “Hello World”. Appends Hello World <p> to the HTML document.`
		`<head>`
		`<script>`
		`function addPara() {`
		`var newPara = document.createElement(“p”);`
		`var newText = document.createTextNode(“Hello World!”);`
		`newPara.appendChild(newText);`
		`document.body.appendChild(newPara);`
		`}`
		`</script>`
		`</head>`
		`<body onload=“addPara()”>`
		`</body>`

`Arrays`	Created by declaring the array elements in [ ]. An array can be assigned to a variable, usually using the keyword const or var. Arrays use zero based indexing to access their elements.	`const Beatles = [“Ringo”, “Paul”, “George”, “John”];` `//Here Beatles[0] is “Ringo”.`

`Date()`	Constructor is new Date([optional parameters]). If the constructor is declared with no parameters, it returns current local date and time. New dates can be created by passing parameters to new Date function.	`//create a new date from a string`
		`var newDate = new Date(“2021-1-17 13:15:30”);`
		
		`//create a new date instance representing 17 Jan 2021 00:00:00`
		`//note that the month number is zero-based`
		`var newDate = new Date(2021, 0, 17);`

`document.createElement()`	Takes one tag name parameter and creates an element with that name. Can place the element elsewhere on the page using functions like insertBefore(), appendChild(), replaceChild().	`//Creates the element <p> and text “Hello World”. Appends Hello World <p> to the HTML document.`
		`<head>`
		`<script>`
		`function addPara() {`
		`var newPara = document.createElement(“p”);`
		`var newText = document.createTextNode(“Hello World!”);`
		`newPara.appendChild(newText);`
		`document.body.appendChild(newPara);`
		`}`
		`</script>`
		`</head>`
		`<body onload=“addPara()”>`
		`</body>`

`document.createTextNode()`	Takes a string as input text and returns a text node with the input text.	`//Creates the element <p> and text “Hello World”. Appends Hello World <p> to the HTML document.`
		`<head>`
		`<script>`
		`function addPara() {`
		`var newPara = document.createElement(“p”);`
		`var newText = document.createTextNode(“Hello World!”);`
		`newPara.appendChild(newText);`
		`document.body.appendChild(newPara);`
		`}`

		</script> </head> <body onload=“addPara()”> </body>
document.getElementByID()	A method of the DOM that takes an ID value parameter and returns an element that matches the id.	//Changes the content of the div to “Hello World!” Hello Hello <script> document.getElementById(“div1”).innerHTML = “ Hello World! ”; </script>
document.getElementsByTagName()	A method of the DOM that takes a tag name parameter and returns an array called “NodeList” that contains elements with the specified tag name.	//Gets an array of all elements in a document with the tag. var tagNameArray = document.getElementsByTagName(“p”);

document.write()	Writes HTML or JavaScript to a document. Note that it overwrites any other text in the document so is mostly used for testing purposes only.	//Writes “Hello World” to the output stream.
		document.write(“Hello World”);

element.getAttribute()	Returns the value of the specified attribute. Takes one parameter: the attribute name whose value is to be returned.	//Removes the CSS style color blue <script> var div1 = document.getelementById("div1").getAttribute(“style”); </script>
element.innerHTML()	A property of the Element class that returns or alters contents of an HTML element as a text string.	//Changes the content of the div to “Hello World!” Hello Hello <script> document.getElementById(“div1”).innerHTML = “ Hello World! ”; </script>
element.removeAttribute()	A property of the Element class that removes all previously set inline CSS styles for a particular element. Takes one parameter: the attribute name that is being removed.	//Removes the CSS style color blue <script> var div1 = document.getelementById("div1").getAttribute(“style”); </script>

element.setAttribute()	A property of the Element class that overwrites all previously set inline CSS styles for a particular element. Takes two parameters: the attribute name that is being set and the attribute value the attribute is set to.	//In all elements named “theImage” sets the name of all src attributes to “another.gif”
		document.getElementById(“theImage”).setAttribute(“src”, “another.gif”);

element.style()	A property of the Element class that returns or alters inline CSS. Syntax is element.style.propertyName = value	//Changes the CSS style color from blue to red
		
		<script>
		var div1 = document.getelementById("div1");
		div1.style.color = "red";
		</script>

Error Objects	Instance creates two properties about the error: message that contains description of the error and the name property identifies the type of error. Generic error plus 6 other core errors: TypeError, RangeError, URIError, EvalError, ReferenceError, SyntaxError.
	Error object can be extended to create custom error messages using the throw keyword.
		//Catch statement defines a block of code to be executed if an error occurs in the try block.
		catch (err) {
		document.getElementById(“myfile”).innerHTML = err.name;
		}
		//Creates custom error message
		throw new Error(“Only values 1-10 are permitted”);

History Objects	The history object is part of the window object and contains the URLs visited by the user within a browser window. It exposes useful methods and properties that let you navigate back and forth through the user's history and manipulate the contents of the history stack.	//Go back two pages if the history exists in the history list.
		history.go(-2);

insertBefore()	An HTML DOM method that, after creating an element, places a child element in the appropriate location before an existing child. The method takes two parameters, the node object to be inserted and the existing node to insert before.	//Creates a new element and places it in the elementList before the first child of
		let newLI = document.createElement("li");
		newLI.innerText = "new Element";
		let elementList = document.getElementById("thisList");
		elementList.insertBefore(newLI, elementList.childNodes[0]);

Location Objects	The location object is part of the window object and contains information about the current URL.	//Returns the hostname property
		let myhost = location.hostname;
		newLI.innerText = "new Element";

Navigator Objects	The navigator object is part of the window object class in the DOM that represents the client Internet browser, also called the user agent. There is no standard for this object so what it returns differs from browser to browser.	//Retrieves the name of the browser var browsername = navigator.appName;
onload()	A DOM event that starts a method when a page is loaded.	//Executes myFunction after MyHTMLPage has been loaded document.getElementById(“MyHTMLPage”).onload = function () {myFunction};

replaceChild()	After creating an element, this function replaces a child node with a new node.	//Creates a new node and replaces the second element in “thisList” with the word “blue”
		let secondBullet = document.createTextNode(“blue”);
		var myList = document.getElementById(“thisList”).childNodes[1];
		myList.replaceChild(secondBullet, myList.childNodes[1]);

Screen Objects		//Returns the height and width of the user’s screen

	The screen object is part of the window object class in the DOM that can be used to return properties about the user's screen.	`var height=screen.height;` `var width=screen.width;`
`Window Objects`	The DOM window object is at the top of the DOM hierarchy and serves as the global object. Everything in the DOM takes place in a window. The window object controls the environment that contains the document.	`//Opens a new browser window with the specified URL` `window.open("http://www.w3schools.com");`
`window.open()`	Opens a new window. The first parameter is a path, a URL, or an empty string, and optional parameters include the window name, features such as the placement of the window or the dimensions, and a Boolean replace value. The feature parameter is a comma separated string of name-value pairs and the replace parameter is an optional Boolean. This parameter has been deprecated so modern browsers may not support it. This method returns a reference to the new window object.	`//Opens a new window that opens the IBM home page and has a width of 600 and a height of 800)` `let thisWindow = window.open("http://www.ibm.com", "myWindow", "width"=600, "height"=800);`
`window.scrollTo()`	Scrolls to a particular place in a window. Parameters include the x-coordinate which is the left-most pixel and the y-coordinate which is the upper-most pixel.	`//Scrolls the window to the pixel located at the coordinate (20, 200)` `window.scrollTo(20, 200);`

`Wrapper Objects`	Primitive types can be converted to objects using wrapper objects. They are the same name as the primitive except they start with uppercase letter. The `typeof` keyword returns a string indicating the data type of the operand.	`//Enables the use of properties and methods of the String class such as the property n.length`
		`let n = new String ("abc");`
		
		`//Returns string`
		`typeof "abc";`
		
		`//Returns object`
		`typeof new String("abc");`

Changelog

Date	Version	Changed by	Change Description
25-10-2022	1.0	Michelle Saltoun	Initial version created