Module 4 Cheatsheet: JavaScript Programming for Web Applications

Class or Method	Description	

`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> a`
		`Hello World <p> to the HTML`
		`<head>`
		`<script>`
		`function addPara() {`
		`var newPara = document.c`
		`var newText = document.c`
		`newPara.appendChild(newT`
		`document.body.appendChil`
		`}`
		`</script>`
		`</head>`
		`<body onload=â€œaddPara()â€`
		`</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 = [â€œRingoâ€`
		`â€œJohnâ€];`
		`//Here Beatles[0] is â€œRin`

`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`
		`var newDate = new Date(â€œ2`
		
		`//create a new date instanc`
		`//note that the month numbe`
		`var newDate = new Date(2021`

`document.createElement()`	Takes one tag name parameter and creates an element with that name. Can place the element	`//Creates the element <p> a`
		`Hello World <p> to the HTML`
		`<head>`
		`<script>`

	using functions like insertBefore(), appendChild(), replaceChild().	var newPara = document.c var newText = document.c newPara.appendChild(newT document.body.appendChil } </script> </head> <body onload=â€œaddPara()â€ </body>

document.createTextNode()	Takes a string as input text and returns a text node with the input text.	//Creates the element a
		Hello World to the HTML
		<head>
		<script>
		function addPara() {
		var newPara = document.c
		var newText = document.c
		newPara.appendChild(newT
		document.body.appendChil
		}
		</script>
		</head>
		<body onload=â€œaddPara()â€
		</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 th <div id=â€œdiv1â€ Hello Hello <script> document.getElementById(â€œ World! â€; </script>
document.getElementsByTagName()	A method of the DOM that takes a tag name parameter and returns an array called â€œNodeListâ€ that contains elements with the specified tag name.	//Gets an array of all elem tag. var tagNameArray = document

`document.write()`	to a document. Note that it overwrites any other text in the document so is mostly used for testing purposes only.	`//Writes â€œHello Worldâ€ document.write(â€œHello Wor`
`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 col <div id="div1" style="color <script> var div1 = document.getelementById("div </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 th <div id=â€œdiv1â€> <p>Hello</p> <p>Hello</p> </div> <script> document.getElementById(â€œ World!</p>â€; </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 col <div id="div1" style="color <script> var div1 = document.getelementById("div </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 â€œ attributes to â€œanother.gif document.getElementById(â€œ â€œanother.gifâ€); `

element.style()	A property of the Element class that returns or alters inline CSS. Syntax is element.style.propertyName = value	<div id="div1" style="color <script> var div1 = document.getele 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
		error occurs in the try bloc
		catch (err) {
		document.getElementById(â€
		}
		//Creates custom error mess
		throw new Error(â€œOnly val

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.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	//Creates a new elemen before the first child of <u let newLI = document.create newLI.innerText = "new Elem let elementList =Â document elementList.insertBefore(ne

	before.	

Location Objects	The location object is part of the window object and contains information about the current URL.	`//Returns the hostname prop`
		`let myhost = location.hostn`
		`newLI.innerText = "new Elem`

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` `var browsername = navigator`

onload()	A DOM event that starts a method when a page is loaded.	`//Executes myFunction after`
		`document.getElementById(â€œ`
		`{myFunction};`

replaceChild()	After creating an element, this function replaces a child node with a new node.	`//Creates a new node and re`
		`â€œthisListâ€` with the word
		`let secondBullet =Â documen`
		`var myList =`
		`document.getElementById(â€œt`
		`myList.replaceChild(secondB`

Screen Objects	The screen object is part of the window object class in the DOM that can be used to return properties about the userâ€™s screen.	`//Returns the height and wi`
		`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 windo` `window.open(â€œhttp://www.w`
window.open()	Opens a new window. The first parameter is a path, a	`//Opens a new window that o` `width of 600 and a height of`

	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.	`a€œmywindowa€, a€œwidtha€=`
`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 (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	`//Enables the use of proper such as the property n.lengt`
		`let n = new String (â€œabcâ`
		
		`//Returns string`
		`typeof â€œabcâ€;`
		
		`//Returns object`