Still struggling with the difference between iterable objects and non-interable..

Assignment Notes
callback function - a single function that expects to operate on one item at a time.
`Symbol.iterator` method
`Array.from--- const makeArray = Array.from (data to be array,`
Iterable object: An iterable object is any object that has a Symbol

```
Interator:
forEach() {}.  forEach (item—could be array, list, function.)
map(){}
filter(){}
reduce() {}
```

Chapter 5- Objects

OBJECT LITERALS

Objects are self-contained set of related values and functions. They keep related information and functionality together in the same place.

Methods …if the property's value is a function

Object literal is an object that is created directly in the language by wrapping al its properties and methods in curly braces.

Objects are mutable at any time.

To create--- const *objectName* = {};--object literal notation or *objectName* = newObject();

To access properties of an object….use dot notation. *objectName.property.* superman.name
Or by using brackets *objectName['property']* –brackets allow you to access nonstandard names and use it as the property key.

If a property doesn't exist, undefined is returned.

COMPUTER PROPERTIES

You can use operators to return the value of the code using it. The '+' is used fro concatenation.

Ternary operator--→ *condition* ? *if false then do* : *if true then do*

Symbol date type can also be used as a computed property key.---more on Symbol data type

--const name = Symbol('name');---"name" becomes the property key.
To add new property using symbol..use brackets. Const realName = Symbol('real name');

--supergirl[realName] = ‘Kara Danvers’;

Symbols can be reused for other properties. Each symbol is a unique value.

CALLING METHODS

Can use dot or bracket notations…objectName.method() or objectName[‘Method’]()

CHECKING IF PROPERTIES OR METHODS EXIST

Use “in” to see if the property exists…’property’ in objectName—returns Boolean or can use objectName.property !== undefined;--returns Boolean too.—all return if inherited from another object.

--hasOwnProperty();---objectName.hasOwnProperty(‘property’);--returns Boolean too.

FINDING ALL THE PROPERTIES OF AN OBJECT
--use a for in loop to find all the objects properties and methods---*for* (const varName *in* objectName)
--console.log(varName + “: “ + objectName[varName]);…varName is just name for variable.

--Object.keys() method—returns an arraow fo all the keys of an object…*for* (const varName *of* Object.keys(objectName)) {console.log(key);}

--Object.entries() returns an arrawy of key-value pairs….in an array—to access:
For(const [key,value] of Object.entries(objectName)){
Console.log(key + “: “ +value);

ADDING PROPERTIES

Add a new property or method by simplying assigning a value to the property.
objectName.propertyName = ‘valueName’;

objects are not ordered lists like array, set, or map.

CHANGING PROPERTIES

You can change the an object property any time through assignment. objectName.propertyName = ‘newValue’;

REMOVING PROPERTIES
Delete…. Delete objectName.PropertyName

NESTED OBJECTS—when objects contain ther objects. The values in nested objects can be accessed by referencing each property name in order using either dot or bracket notation:

objectName.nestedObjectName.PropertyName. or
objectName[‘nestedObjectName’][propertyName]—or mix them up.

COPY OBJECTS BY REFERENCE
--const oldObjectName = {propertyName: ‘valueName’};….then const newObjectName = oldVarName;…then the properties are all the same and change in both objects.—this does not happen iwht primative values…numbers, strings, etc.

OBJECTS AS PARAMETERS TO FUNCTION
Object literals can be passed as a parameter to a function….

Object Literal-- an object literal is a comma-separated list of name-value pairs inside of curly braces. Those values can be properties and functions.---function funName({propertyName1 propertyName2, propertyName3}); then
Execute like this--funName({propertyName1: value1, propertyName1: value2, propertyName3: value3});--values were assigned to the arguments.---named parameters-used when there are large amounts of optional parameters.

--this

The keyword “this” refers to the object that it is within…it is used inside methods to gain access to the object’s properties.

NAMESPACING

Naming collisions occur when the same variable or function name is used for different purposes by code sharing the same scope

Object literal pattern…create an object literal to serve as the namespace.---

```
Const = nameSpace = {
--function1 () {};
Function2(){};
Function3(){};
};
```

Call the functions by nameSpace.function1() or nameSpace.function()2;

BUILT-IN OJECTS
The main built in objects are arrays and functions…but there are other global opjects that can be accessed anywhere.

JSON
A light-weight data storage format.
A string representation of the object literal notation, with a few differences:

--property names must be double-quoted
--permitted values are double-quoted strings, numbers, true, false, null, arrays, and object
--functions are not permitted values
A JSON has methods that can be done in JS.
--parse()—takes a string of data in JSON format and returns a JS object.
JSON.parse(JSONstringName); it will return
{propertyName1: 'valueName1', propertyName2: 'valueName2', propertyName3: 'valueName3',etc}

--stringify() does the opposite…takes JS object and makes JASON data. (can't include functions and are ignored)

You can add spaces between each key-value---JSON.stringify(objectName, null, " ");

The "Math" Object
The Math is built-in objects that has several properties and methods…which are common.

MATHEMATICAL CONSTANTS
--capital letters are convention for constant values
Euler's constant?

MATHEMATICAL METHODS
Several available
ABSOLUTE VALUES—Mathh.abs() returns the absolute of a number.

ROUNDING METHODS
--Math.ceil() rounds up to next integer
--Math.floor() rounds down or remains the same if already an integer
Math.round() rounds to nearest integer.
Math.trunc() returns the integer part of the number….truncated after decimal

POWERS AND ROOTS
-Math.exp() method will raise a number to the power of Euler's constant.
Math.pow(x,y) raise a number (x) to the power of another number(y)
Math.sqrt() returns positive square root
Math.cbrt() returns cube root
Math.hypot()

LOGARITHMIC METHODS
Math.log returns natural logarithm …Math.log2() for base 2 math or Math.log10() for base 10 math

MAXIMUM AND MINIMUM METHODS

Math,max()—returns the max number of aurgements

TRIGONOMETRIC FUNCTIONS

There are a bunch of Math Object in standard trigonometry functions…including radians (a standard unit of angular measurement.

Rounding errors can be a problem in some programs…but not most. Figure out how close the number needs to be before programming. The Math.PI is especially tricky.

RANDOM NUMBRES

Math.random() is used to create random numbers which can be very useful when writing and programs…it produces a number between 0 and 1 such as .788888. To generate a number between 0 and x.   multiply the random number by x. …x * Math.random

THE DATE OBJECT

Date objects contain information about dates and tiems. They all represent a single moment in time.

CONSTRUCTOR FUNCTION

A contructor function is used to create a new date object using the new operator

Const today = newDate();
--today.toString(); ---produces the date in a string format.

`new Date(year,month,day,hour,minutes,seconds,milliseconds)—parameters can be used.`

Computer starts with 0---so 0=January, 1=February
Epoch is 1st January 1970—used to make calculating dates easier.

GETTER METHODS

Methods that are used to return data from the Date object. But the date object first has to be created…will return month, year, or day, etc.

UTC is the primary time standard by which the world regulate clocks.
--getDay(), getUTCDay(), getDate(), getUTCDate(), etc…Month, FullYear, Hours, Minutes

getTime() returns a timestamp from the Epoch date.
--getTimezoneOffset() returns the difference in minutes between local and UTC>

SETTER METHODS

They return a timestamp of the undated date object…returns the timestamp from the Epoch date and need to have a .toString()…setDate(7), setMonth(10), setFullYear(2018)= Nov 07, 2018

THE REGEXP OBJECT

A pattern that can be used to search strings for matches to a pattern.

CREATING REGUAL EXPRESSIONS

1st way and preferred—use literal notation of writing the regular expression between forward slashes. –const pattern = /[a-zA-Z+ing$/;

2nd -- `const pattern = new RegExp('[a-zA-Z]+ing');--`you can use strings with this method.

REGEXP METHODS

Use test() to see if the string matches the regular expression pattern. Returns Boolean.—or use exec() to see the array contained or "null" if there is nothing.

BASIC REGULAR EXPRESSIONS

Example const pattern= /JavaScript?;

CHARACTER GROUPS

--Placed inside brackets can be anything inside the brackets…const group1= /[xyz]/. Can be x or y or z.

--Squence of characters…/[A-Z] or /[0-9]/

--groups can be combined for more complex patterns. Ex.

```
pattern = /[Jj][aeiou]v[aeiou]/;
<< /[Jj][aeiou]v[aeiou]/
pattern.test('JavaScript');
<< true
pattern.test('jive');
<< true
```

REGULAR EXPRESSIONS PROPERTIES

Regular expressions are ojects that have the following properties…

--(g)global—returns all matches instead of just the first.

--(i)ignoreCase—patterns are case-insenstive—change to false by redefining the pattern without the "I"

--(m)multiline—makes the pattern multiline and not stop at the end.

You can place the letters after the pattern and / to set the property as true.   Exp. Pattern = /java/I; pattern.test ('JavaScript') = true

SPECIAL CHARACTERS

. –any character expept line breaks, /w any word character, /W any non-word character, /d digit character (numbers), /D any non-digit characters, /s matches any white space. /S macthces any non-whitespace.

MODIFIERS—placed after tokens to deal with multiple occurrences.
? makes the expressional optional, *matches one or more occurrances of the preceding tokens, +matches one or more occurances of the preceding token, ^ makes the position before the first character in the string.. $ marks the position after the last character of a string.
Difference between the "*" and the "+" modifier?

Use backsplash\ to match the character. So for ? use \?

GREEDY AND LAXY MODIFIERS
The examples above are greedy and will match the longest possible string….to make them lazy, add a ? after the modifier.

STRING METHODS
Split()—splits a string into separate elements
--match() returns an array of all the matches instead of just the first one.
--the "g" flag returns all the matches….what does the "g" flag do that is different than just match()
--search()—returns the position of the first match, if it returns -1 then there is not match.
--replace() replaces any match with another string.

MATCHED GROUPS

--ascapturing groups—sub-patterns placed in the expression in ()

What is the difference between forEach and a for…of or

CHAPTER 6

The Document Object Model
You can access elements of a web page and enable interaction.—a HTML document as a network of connected nodes. Everything is a node, the HTML tag is the parent node. DOM is not part of js because it can be used by other programmable languages.

History of the DOM
DHTML—dynamic HTML was first used.—legacy DOM or level 0
DOM specification is a developed as a living standard…growing, changing, and improving.

An Example Web Page
Whitespace is stored as a nodes in HTML

Getting Elements
The methods will return a node object or a node list which is like an array-like object…They can then be assigned a variable.
<body>---document.body—becomes an object now. Use the

-body.nodeType; to find the type of node. 1=element, 2=attribute, 3= text, 8=comment, 9=body
--body.nodeName to find the name of the element "BODY" returned in uppercase
Legacy DOM Shortcut Methods
Document.body---returns body element
Document.images—returns node list of images
Document.links—list of <a> and <area>…that have href attribute
Document.anchors—node list of <a> with a *name* attribute
Document.forms—list of forms

Nodes are not actually arrays but can use length property and index numbers. You can turn node lists into arrays with Array.from method or spread operator…const listArray = Array.from(document.images) or const listArray = […document.imgaes];

--document.getElementByID(elementName);
--document.getElementByTagName('li');
--document.getElementsByClassName('className');
--document.querySelector()—uses CSS notation--- ('#IdName' or '.className')
--document.querySelectorAll();--uses CSS notation--- ('#IdName' or '.className')
Other examples

```
const wonderWoman = document.querySelector('li:last-child');
const ul = document.querySelector('ul#roster');
const batman = ul.querySelector('li#bats')
```

Navigating the DOM tree

childNode----nodeList.childNodes—will return all the child nodes of that node object…including text and whitespace in the text.
--nodeList.children --only returns element nodes.
-- nodeList.firstChild
--nodeList.lastChild
.nextSibling, previousSibling
Sometimes they return empty text.

textNode.nodeValue—once. You have referfenced a node, you can use the nodeValue method.
textNode.textContent or .innerText for Explorer 8.

Setting An Element's Attributes

getAttribute()—variableName.getAttribute('class'); or ('src')
setAttribute()---variableName.setAttribute(attribute, newValueofAttribute)…can use to change or add a new attribute Value and attribute.

Can use dot notation…wonderWomen.id;
Don't use "class" and "for" for attribute names.

variableName.className---.className can be used to find the value of the class attribute---use variableName.className = "newClassName"—use with caution can override other names.

variableName.classList.add to list classes of elements and then add an element without overriding.
variableName.classList.remove (”className”);
variableName.classList.toggle (‘className’);--false for removed, true for added;
variableName.classList.contains (‘className’);

classList is not available in older Explorer versions, use functions instead.

Creating Element

--createElement(‘element’)…ie. (‘li’) (‘p’)---the element is empty.
--document.createTextNode()---const variableName = document.createTextNode(‘textForNewElement’);

--createdElementVariableName.appendChild(createdTextNodeName);

Process:

1. Create the element node
2. Create the text node
3. Append the text node to the element node

Add text without appending….const createdElementName.textContent
---.textContent ---to add text to the created element.

You can use Functions to Create Elements

Add Elements to the page with
appendChild()
insertBefore(newNode, nodeToGoBefore);---called on the parent node.
instertAfter()
removeChild()—parentElement.removeChild(elementToRemove);
replaceChild()—parentElement.replaceChild(newText, oldText);

.innerHTML---returns the child element of an element as a string of HTML—returns all the raw HTML. It’s writable and can replace. You can enter raw HTML as a string. Element.innerHTML = ‘html code’;

Updating CSS

You can use the style property to change CSS inside an element.
Can you change it in the CSS document?

Change dashes to camel case…background-color is backgroundColor in JS…ex
element.style.backgroundColor. or element.style[background color’] = ‘ble’;
Display…can make elements appear and reappear. elementName.style.display-‘none’;…or ‘block’ to reappear.

getComputedStyle() works only in inline styles

Use CSS sparingly…best to use the stylesheet when possible.

Quiz Ninja Project

---for in loops through property names (**key**)---for (let = variableName in objectName) {statements}---innumberable objcets but not iterable—will return **index number for arrays and user defined properties(**the key)---to call each value use variableName[objectName]

--for of loops through property **values**. for (variableName of Object) {statements}---for interable objects—passes the **values** of the array.

What happens when only two of the three arguments are given in the function call?

Event Listeners

Event listeners let you know when the event happens so the program can respond.
Blocking approach is when a program has to keep checking for the event instead fo running the rest of the program.

Inline Event Handlers
--document.body.addEventListener("click", doSomething);
"The click event happens when a user clicks with the mouse, presses the Enter key, or taps the screen, making it a very useful all-round event covering many types of interaction."
Inline Event Handler—<p onclick="console.log(you clicked me!)">Click Me</p>

Document.onclick = function(){console.log('you clicked on the page'):}—old method using properties. Still workds but only one function per event.

Event listeners can initiate multiple functions and is the preferred way.
--addEventListener()---document.body.addEventListener('typeOfEvent', function);--when used without node it is global—

The function can be written in the "function space" or reference one.

Example Code—see code

The Event Object
Whenever an event handler is triggered by an event, the callback function is called. This function is automatically passed an event object as a parameter that contains information about the event.

--event handler is "Event handlers are **the JavaScript code that invokes a specific piece of code when a particular action happens on an HTML element**. The event handler can either invoke the direct JavaScript code or a function." Something that invokes a function or code.

Event Object---

Removing Event Listeners

Types of Events

.type property tells the type of event such as "click"
.target property tells which node fired the event.

Corridnates of an Event—screenX and screen—number of pixels from left top of screen
--.clientX and .clientY show left and top of the client (like browser)
.pageX and .pageY show document coordinates.

Mouse events
Click, mousedown mouseup events

```
const clickParagraph = document.getElementById('click');
clickParagraph.addEventListener('click',() => console.log('click') );
clickParagraph.addEventListener('mousedown',() => console.log('down') );
clickParagraph.addEventListener('mouseup',() => console.log('up') );
```

'—dbclick-double click
--mouseover—when poiner is placed on element.
--mouseout-when pointer moves away

Keyboard events
--keydown—when key is pressed
--keypress—after keydown and before keyup
--keyup—when key is released

Modifier Keys'
Shift ctrl alt meta fire on keydown and keyup
.key property to see which key was used.
.shiftKey .ctrlKey, etc. will give a ture statement if that particular key was held down.

```
addEventListener('keydown', (event) => {
if (event.key === 'c' && event.ctrlKey) {
        console.log('Action canceled!');
    }
});
```

Touch Events—used on smartphones, tablets, etc.
--touchstart—when the user initially touches surface.—use carefully
--touchend—when stop touching
--touchmove—after touched and moved around but without leaving.
--touchcenter—when already touched and passes over the element the event listener is attached to.
--touchleave
--touchcancel
Swipe events are done using a combination of event properties.

Touch Event Properties
--.length…events.touches.length---how many touch points are in contact with the surfaces.
.touch.screenX, .screenY, .radiusX .radiusY, touch.force—amount of pressure between 0 and 1.,
touch.identifier

Removing Event Listeners

removeEventListener()…don't use anonymous functions with addEventListener() and removeEvetnListener() bcause you won't be able to reference them later.

Stopping Default Behavior

preventDefault() …used inside a callback function to stop the default behavior.

Event Propagation

When. You click on an event. You are also engaging all the elements that it is nested inside of…the parent node, grandparent node, etc.
Event propagation-
Bubbling when the events fires on element clicked first and then bubbles up through the rest.—default.
Use stopPropagation() to stop the bubbling.

Capturing starts by firing on the root element until it reaches it's target—addEventListener() has 3rd parameter…Boolean—default is false. So to change that add a third parameter:

```
ulElement.addEventListener('click', (event) =>
console.log('Clicked on ul'),true);
```

Event Delegation
Quiz Ninja Project
Chapter Summary

What is the difference between an "array" and an list in an object.

.innerText vs .createNodeText

Questions:

When is better to use .innerText or .createNodeText?

You can change CSS inside an element, what is required to change the actual file?

It seems that the forEach can be used for the same purposes as "for...of" loop,is that correct?

I appear to be getting an error in the "view.hide" function, or at least it is not working on the start button. Since I copied and pasted it, and spent quite a bit of time trying to debug it, I am still not sure what is wrong.