Review

1. Return means enter

Udemy JavaScript Course – (Jonas Schmedtmann)

1. Welcome!
2. JS Fundamentals P.1

**Values and Variable –**

‘Jonas’, 23 = value = basically the smallest unit of information that we have in javascript

firstName = variable

Rules: Use camelCase, if it’s a contant write it in uppercase

**7 Data types** (Numbers, Strings, Boolean, undefined, null, symbol, big int)

Dynamic typing = you don’t need to defined their data type

Use let when you don’t want it to be constant as you cant change it anymore

X += 10 === x = x + 10

We use comparison operators to produce Boolean values

**OPERATOR PRECEDENCE** (also, search on google) – basically just mean what will be the first one to command

I’ve learned how to make a Boolean value, and applied the formula ang getting BMI at the first Code Challenge

**Strings and Template Literals** – backticks `` (for writing template literals)

**\n\** – is a new line

**IF STATEMENTS –** we can also put an undefined variable then give it a condition after what it will be later, and also try the less than equal,

**VALUE TYPES –** converging between types (number to strings)

**NaN –** invalid number

**CONVERTION –** manually we use number(variable), string(value ex. 23)

**COERTION –** automatically behind the scene , addition is dum dum using string coercion conversion

**TRUTHY AND FALSY** – not completely false until we convert it to Boolean (5 falsy 0, ‘’, undefined, null, NaN)

Falsy equal to 0 so 0 is false, if (true) else (false)

**EQUALITY OPERATORS == VS ===** also (!==) strick ver (!=) loose version (we also use number(prompt()) so we will not have an error while we use 3 equal sign.

**BOOLEAN LOGIC –** BRANCH OF COMPUTER SCIENCE USE TRUE AND FALSE TO SOLVE COMPLEX LOGICAL PROBLEMS (**AND , OR ||, &&** ) (not operator !)

**SWITCH STATEMENT –** we use this to for complicated if statement, compare value to multiple options, we need to say to stop or give some break, we also need break

**STATEMENTS AND EXPRESSION (**expression produce value**)** (statement have complete sentence or statement is statement doesn’t really mean to produce value, and all about action), expression and statements are not allowed to mix;

**THE CONDITIONAL (TERNARY) OPERATOR –** (to use emoticon window + .)

**FACT:**

**TERNARY –** use this for only one if and else and also after what’s the only difference between the ternary statement you can use it to the string template literal, then to execute it.

**SWITCH –** use this if you only have one variable to change

**FUNDAMENTALS PART 2**

**ACTIVATING STRICT MODE –**  ‘use strict’; (just like border box), no comment code, good in function, we can easily found errors than without strict mode, somehow we can see the little misspell

**FUNCTION –** piece of code we can use repeatedly (calling / running / invoking function)

To write clean code, don’t repeat yourself or dry

**FUNCTION DECLARATIONS VS EXPRESSIONS –** function declaration difference is you can put execution on first

**ARROW FUNCTION –** (shorter and faster to write) return will automatically automated.

Const (function name) = (parameter name) => (return value) // Function

**FUNCTION CALLING OTHER FUNCTIONS =** the name of the other function could be use inside the fu0nction.

**REVIEWING FUNCTIONS –** function declaration (function that can be used before it’s declared), function expression (essentially a function value stored in a variable), arrow function (great for a quick one-line functions. Has no this keyword (more later)).

1. Need a function name
2. Need a parameter or not (you can use one or more parameters)
3. Then we have variable inside function
4. Later we use that variable to part of execution.
5. Then return the variable.
6. Then we call the execution of function
7. And input data in parameters name by the function name.

**ARRAYS –** Onevariable that uses a lot of value using bracket (arrays are zero based)

**BASIC ARRAY OPERATIONS –** weuse **push** to put value in the end of array, we use **unshift** to put value in the first of the array, we use the **pop** to remove the value in the end of array, we use **shift** to remove the first value of the array

**INTRODUCTION TO OBJECTS –** we can’t put variable the array inside the array. So we have object for that, properties, most fundamental concept in a javascript language,`

**FACT –** use array when more order data, and object for more unstructured data

**OBJECT METHODS –** You can put function as an object value, declaration doesn’t work on object it should be expression

**this –** inside the object of the object where function is place **remember:** you need to execute first the function before you log the input

**LOOP –** One of the other control structures (like if else statements), so we can do repetitive task

**For –** hasa counter (variable and value = value and condition = execution)

**LOOPING ARRAYS, BREAKING, AND CONTINUING –** we canusefor property like break and continue to add on if statement.

**FOR LOOPING BACKWARDS AND LOOPS IN LOOPS –** we use I to indicate the length and then making an statement then execution. And we can also put loop in loop just right before the bracket end put another loop on it

**THE WHILE LOOP –** more versatile than the for loops, (careful with infinite values)

Adding **continue** on if statement will just continue the process and don’t mind the thingy.

**SEVEN (DOM AND EVENTS)**

**DOM MANIPULATION –** document query selector (‘.message’). textContent. (document object model)

**EVENT LISTENER –** event listener then the type of event example of click then the function, eventlistener (‘click’, function (){})

**MANIPULATING CSS-** select the dom first then add the style then the css properties, also you need a value of string. classList add/remove (‘and you don’t need the dot’)

How JAVASCRIPT works behind the scene.

**High-level** – It requires better hardware resources and it also has build in syntax not like C which you have to put variable types and it pretty much slower compare to C like programming stuffs.

**Garbage Collected-** it will automatically clean your ram

**Interpreted or just in time compile –** 0 -1

Multi- **Paradigm –** approach and mindset of structuring code, direct your coding style and technique. 1. Procedural programming, object-oriented programming, functional programming, imperative vs. declarative.

**Prototype-based object-oriented –** almost everything is just object except primitive values.

**First-class function –** we can use function as variable

**Dynamic –** no need for data type definitions, and it will automatically change.

Concurrency model – how Js engines handles multiple task

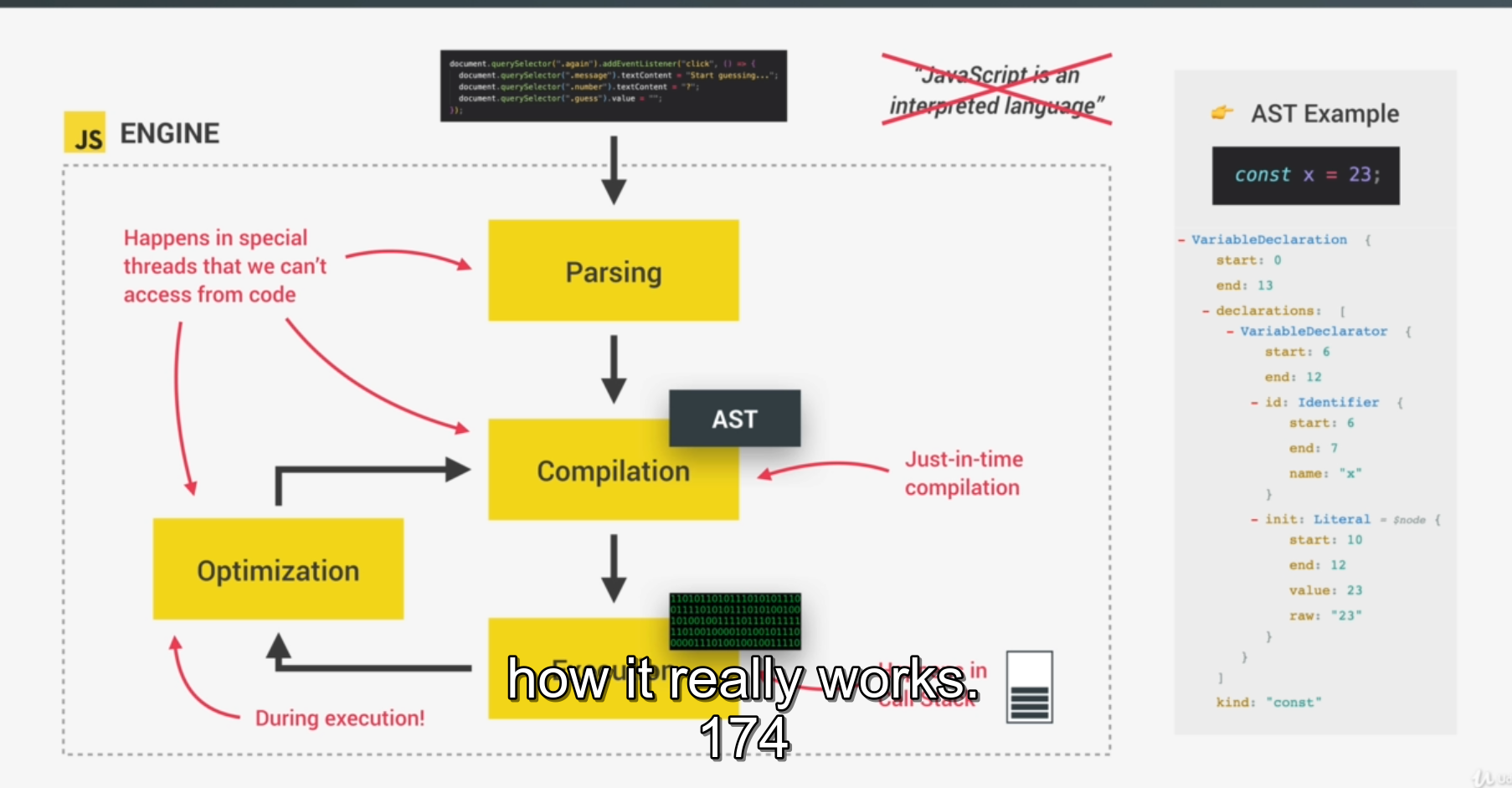
**Single-threaded** – it only can do one thing at a time

**JS ENGINE –** program that executes javascript code. Google v-8 and node.js call stack, where code executed which called execution context/ heap- stores all the objects that our applications need.

**Computer science sidenote: compilation vs interpretation**

**Compilation –** the whole code will be converted into a machine code and written by the binary file and can be executed by the computer, **Interpretation** is runs through source runs line by line.

But **Javascript** now is a mixed of compilation and interpretation



Web API – functionalities provided to the engine, accessible on window project.

Callback queue – click timer data (callback function from dom event listener.

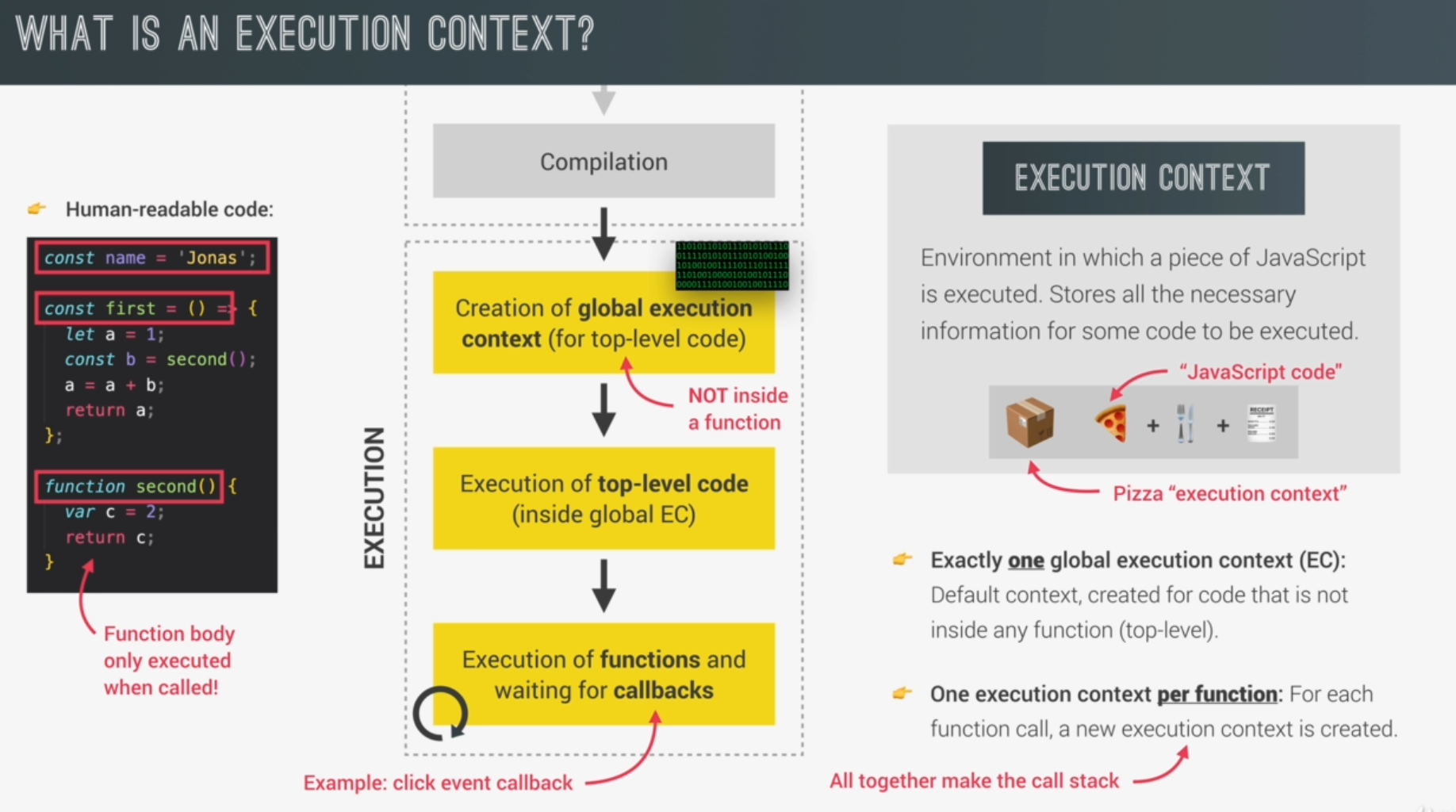
Multiple c++ bindings and thread pool

**How javascript is executed?**

After compilation the global execution context was created (not inside a function)

**Execution context –** it’s just basically a box environment which stored the code to be executed.

**Execution of top level code –**



Arrow functions doesn’t have arguments objects and this keyword.

**SCOPE AND SCOPE CHAIN –** so probably scope are like how you will access the variables of an object, there are global scope, function scope and block scope, which the accessible of their elements is depend of what kind of values they are.

**Scope Chain -** you can access elements by using parent scope,

**Variable Environment Hoisting -**  make variable accessible before they actually declared

**Primitive vs Objects –** Primitive is just basically variable and objects are using a lot of variable example or an array,

PART 9

**Destructuring Arrays – ES6 feature** to break the complex data structure down to smaller data structure like variable, what we do is we const or let then we put an array [a, b, c] is equal to the array, and by doing this we can also make them define separatedly.

**Destructuring Object –** the only difference between array is that you use curl bracket on object and square bracket on array.

**The Spread Operator –** unpacking all elements at one by using the triple dot (…),

**Rest pattern** – same the usage of dots, but to collect multiple elements and condense into array, while spread is to unpack.

**Short Circuiting –** When the first statement is true the other will not even seen, or return the end value if they’re all falsy value. || , && is when get false the other one will not even look at, and the value where it stopped that the value that it will return, whenever the reading makes true, it will print the value out.

**Nullish Coalescing** – 0 and empty string are not falsy values, by using double question mark,

**Looping Arrays (The for-of loop)** – probably the shortest method of for loop, for longer or bigger data. Using **const** first then the short term of array then it will compile things.

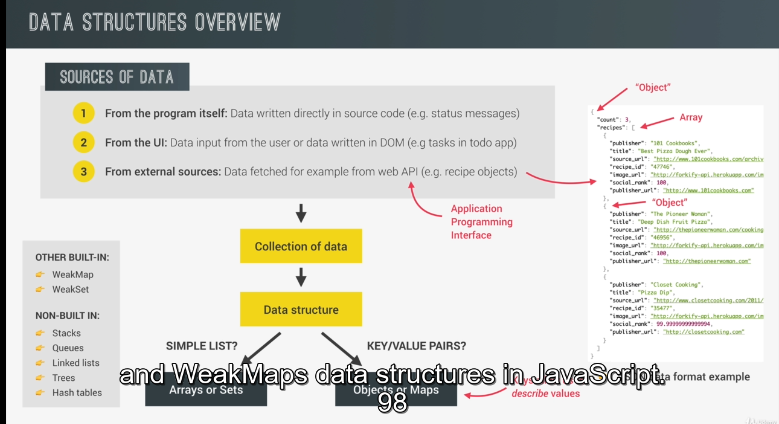
**Enhanced Object-** we now don’t need to put function keyword, and also using the repeated name

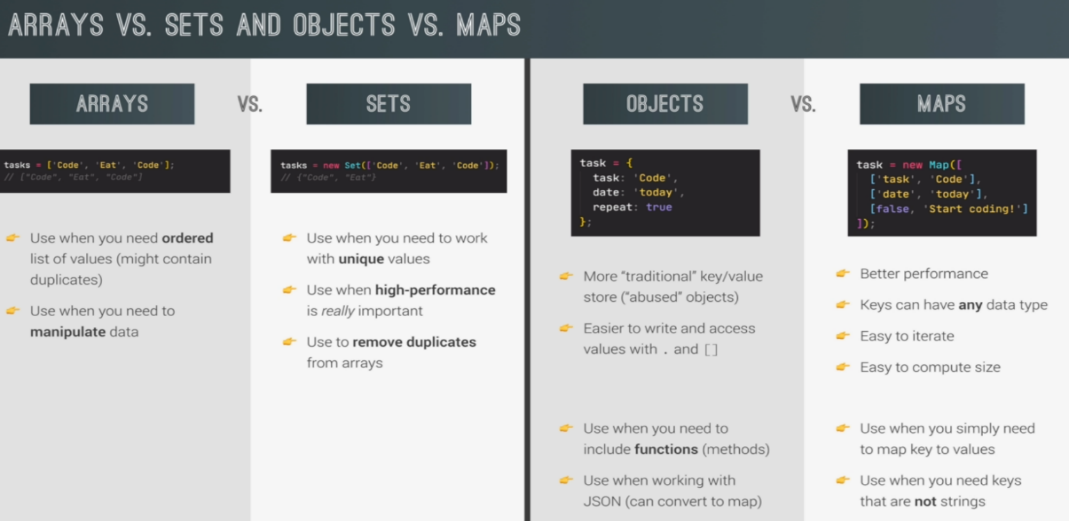
**The Optional Chaining –** we use ?? so it will continue functioning if the value is defined.

**Looping objects –** Using Object.keys and Object.values, looping on an object in the right way, Object.entries make object into an array, we use array of an object by also using looping arrays

**Sets –** Another collection o, there should be no duplicates , using new Set

**Maps –** also using new plus Maps ([[],[]]), good for complicated data **Object.entries** for, **Object.value,** thevariable so called **Object.keys ,** forvalue so called, **.get** for get the keys , .**delete** delete the key, and also **.clear**, **set** is to add new map .**has**

****

****

Split method = using. split (‘ ’) it will create an array, using **join** will make a string, when you need to define the variable next, make sure you have an **empty array**, **replace** keyword will change what you put first inside the close parenthesis to second you enter. **padStart/End(20, ‘+’)** it will target all spaces and take places what you input on second. You can use **.lenght** or **size** if you want, **slice** is for cutting to, and there’s also **trim,** and also .**repeat**, **toLowerCase, toUpperCase**, **replaceAll**, **includes, starts/endWith, includes, lastIndexOf, indexOf**

**Part 10 (A closer look to function)**

**Default parameter –** well you can also make an expression inside the parameter, and declared undefined on skipping function parameter,

**First Class Function** – we have a same reading like value, because functions are objects. But they’re just concepts.

**High Order Function** – function that receives another function as an argument or returns a new function. (Only possible because of first class function)

**Obstruction** – hiding some details in code that ain’t that important. And making a high level of obstruction, you can use **call, apply, and bind**

**Bind** – does not immediately call a function instead return a new function that sent to bound

**Immediately Invoke Function Expression**– function that only wanted to be called once. We can make this thing work by wrapping a function by a parenthesis.

**Closure** – this one is a messed up (laugh), basically you just make the Invoke function call not by once. And also have an access to the variable of its parents even it’s already returned.

**setTimeout** – we use the (function first, then the timer which is converted to millisecond

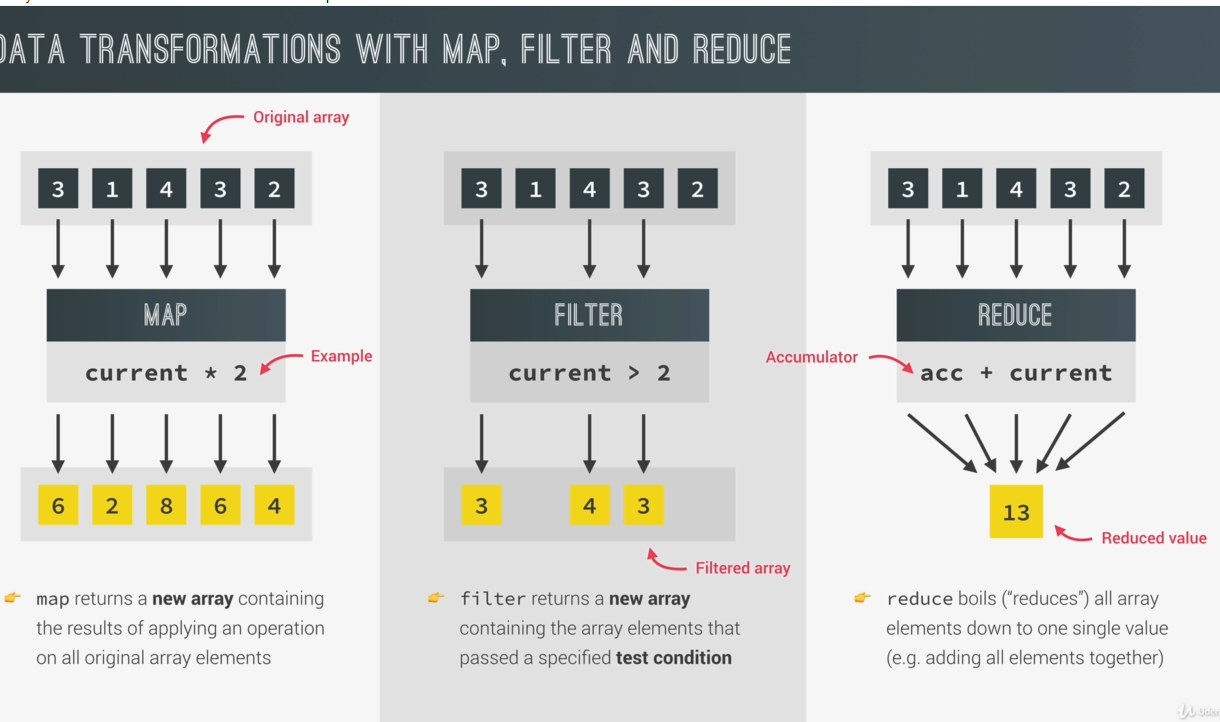
**PART 11 WORKING WITH ARRAYS**

**Simple Array** ­– arrays also have **slice** method like on strings and array also have **splice** method, do this for calling one to another. **Reverse** is the one who reverse, scoping will mutate the arrays

**forEach** **-**. Is higher order function, that requires a callback function.

**Creating DOM Elements** – we use **insertAdjacentHTML (‘afterbegin’, html)** search mdn documentation for more usage. And what this does is just inserting the variable inside the targeted, like inserting some code with format of html to javascript back to html index.

**Data Transformation map, filter, reduce** -



**MAP METHOD** – will return a new array.

**FILTER –** separate or basically filter variables in an array.

**REDUCE –** will make array to one variable

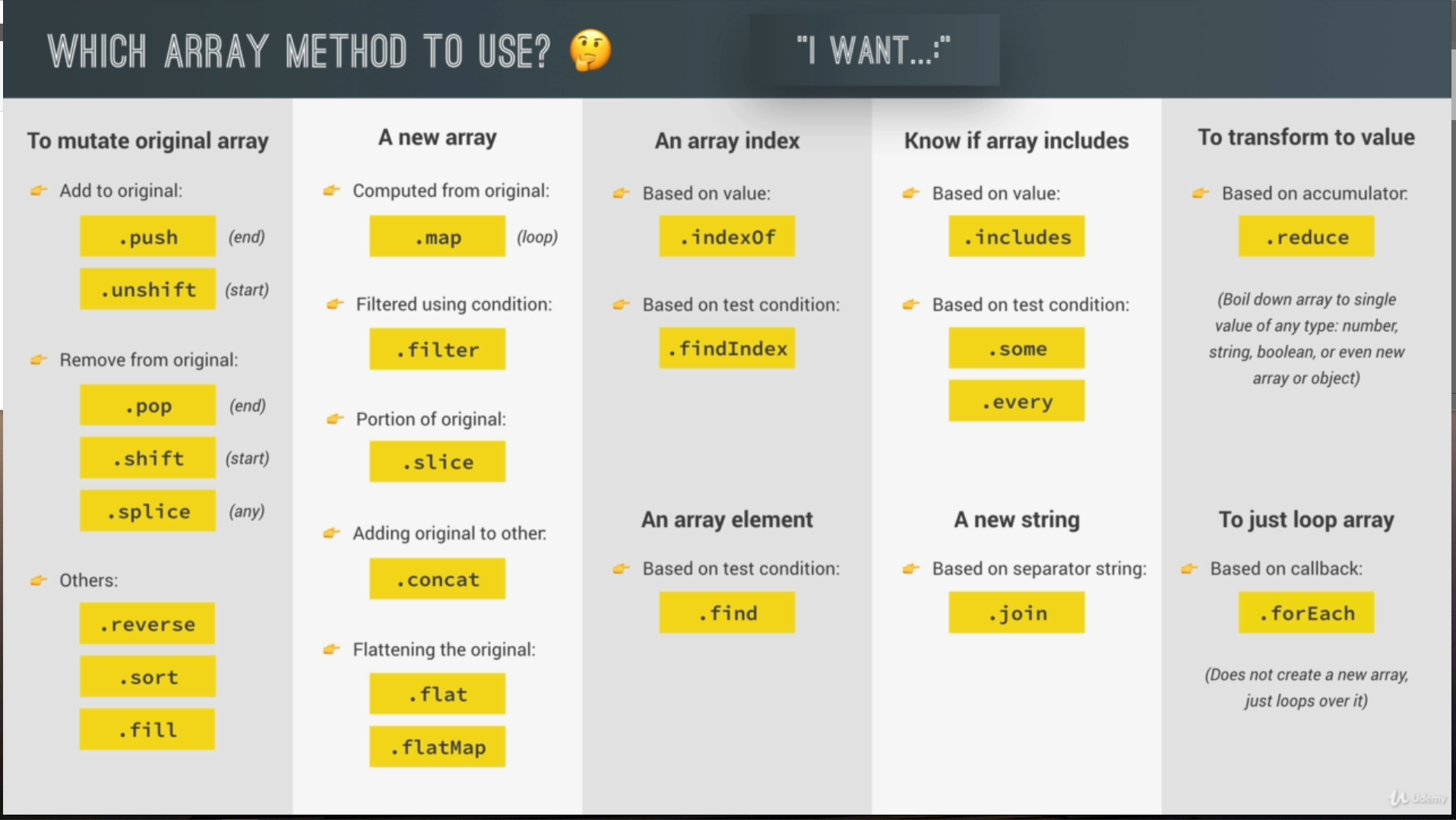
**FIND –** literally will find a value

**findIndex –** find a value

**some and every –** includes have some alternative property using a call back function called as **some** it will return a Boolean value (if there’s one thing that satisfied the condition), **every** will return true if it all values satisfied condition.

**Sorting Arrays** – sort is a string sense, that will return and make an alphabetically array also using (a - b)

Another way of creating array – **fill** (literally will fill the array), **Array.from()** which return much cleaner than the new Array().



Math.trunc() – will remove decimals

**Part 13 – Number, Dates, Intl, Timer**

**Converting and Checking Numbers** - **parseInt** , will convert string to number, **parseFloat**, isNaN, **isFinite, isInteger**

**Rounding** – **sqrt**, **max**, **min**, **round, ceil, floor, trunc, toFixed**

**Remainder Operator –** 5 % 2,

**Working with BigInt –** (2 \*\* 23 - 1) / Number.MAX\_SAFE\_INTEGER (but we can use (bigNumber)n or BigInt(num))

**Creating Dates** – new Date() , .getFullYear/Month/Date/Day/Hours/Minutes/Seconds/Time() , .getTime(), .setFullYear()

**Operation With Dates** – we use the milliseconds to calculate dates

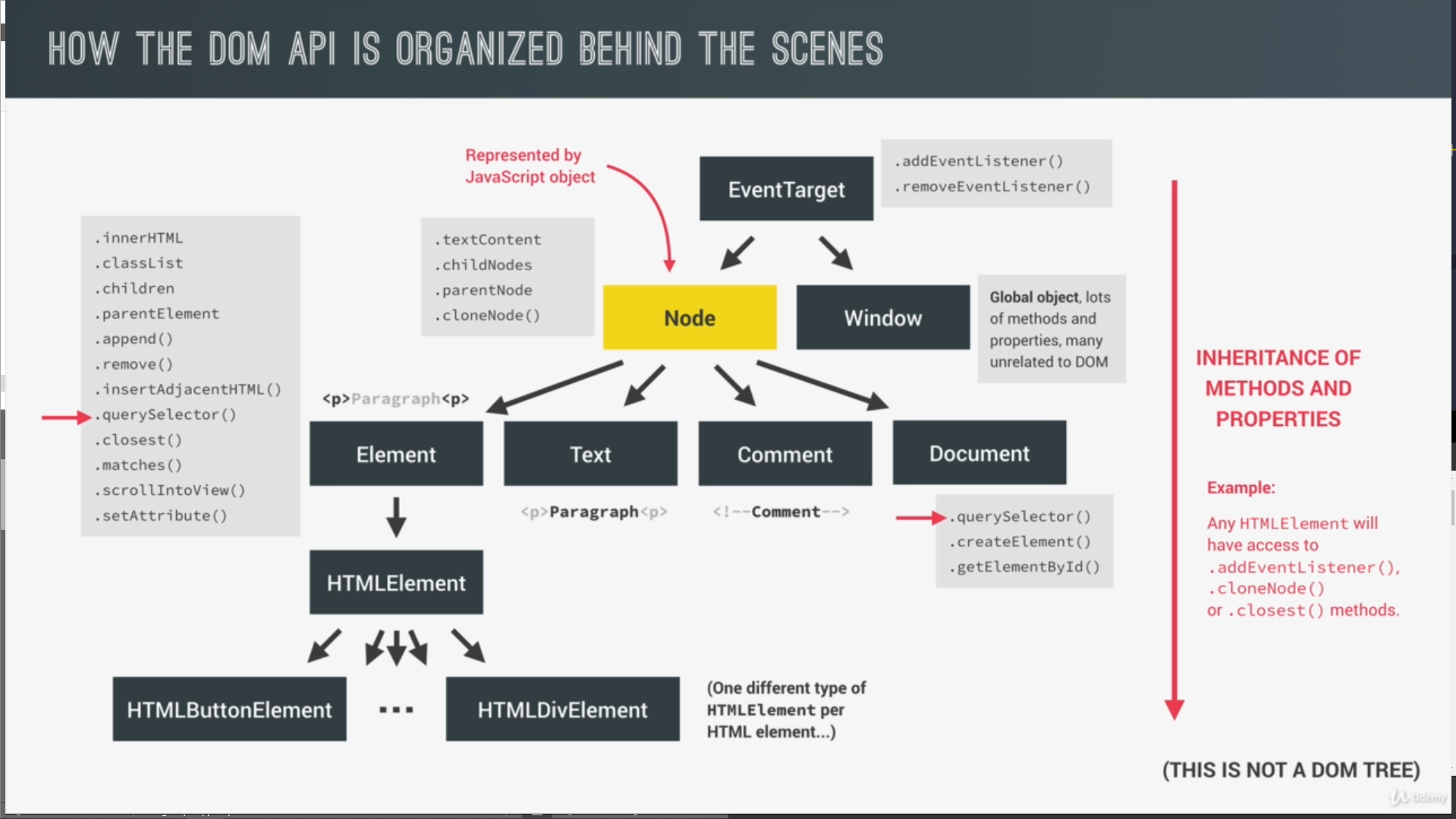
**Internationalizing Dates (Intl)** – iso language code table, Intl.DateTimeFormalt(locale).format(date), mdn intl

**Internationalizing Numbers (Intl)** – formatting number based on language

Timers **setTimeout** and **setInterval** (callback function) clearTimeout

**PART 14 – Advanced DOM and Events Manipulation**

**DOM –** interface between all Javascript code and the browser



**Selecting, Creating and Deleting Elements**

document.**documentElement**/head/body

document.querySelector(‘.header’)/.querySelectorAll

document.getElementById(‘section—1’)

document.getElementsByTagName // all buttons

.insertAdjacentHTML

.createElement

.classList.add

header.prepend(message) // top

header.append(message) // bottom

header.before/afer(message)

message.remove

**Styles, attributes and Classes**-

getComputedStyle(mesaage).color

.setProperty(‘class name’, ‘color’)

.set/getAttribute(‘company’, ‘Bankist’)

**Implementing Smooth Scrolling** – getBoundingClientRect()

window.scrollTo(variable.left + window pageX/YOffset)

window.scrollTo({left: value.left +window.pageX/YOffset, behavior: `smooth`,

section1.scrollIntoView({behavior: ‘smooth’});

**Types of Events and Event Handlers**-

removeEventListener h1.onmouseenter = function (e){}

**Event Propagation Bubbling and Capturing**

Route from parents not to siblings

e.stopPropagation()- stop bubbling

**Event Delegation -** to prevent (using e.target + getAttribute then + scrollIntoView + behavior = smooth

**DOM Traversing** – down ward upward sideways dom traversing

childNodes, children, firstElementChild, lastElementChild, parentNode, parentElement, closest, previous/nextElementSibling,

**Building a Tabbed Component** – you can use **guard classes** it will do things fast, and tabbing using close

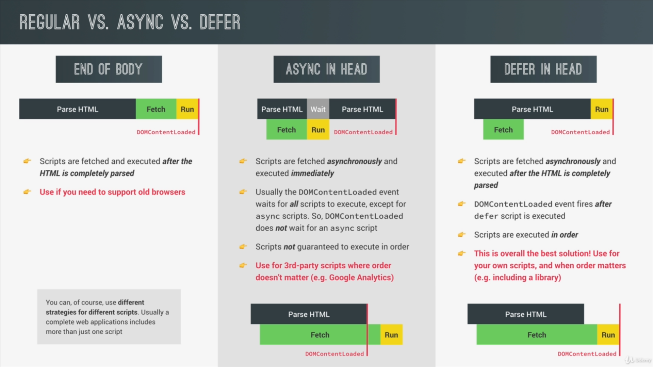
**Passing Argument to Event Handlers** – mouseover, mouseenter, mouseout, adding event listener should have function next to it(well this will not work straight forwardly) we can make an empty function first then proceed to expression,

**Implementing Sticky Navigation –** don’t forget getBoundingClientRect(), its property will give some good values, you can use later.

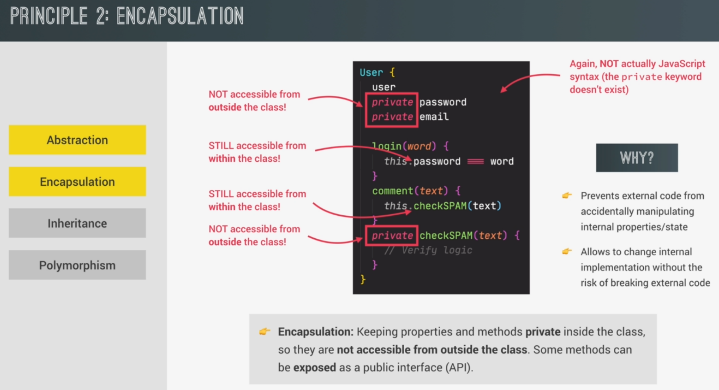
**Intersection Observer –** using new IntersectionObserver, and threshold, root, isIntersecting

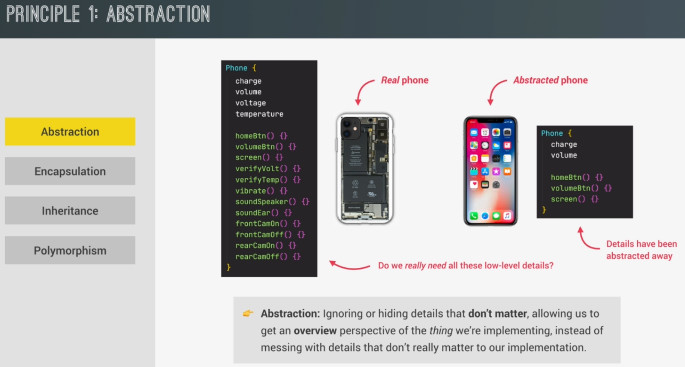
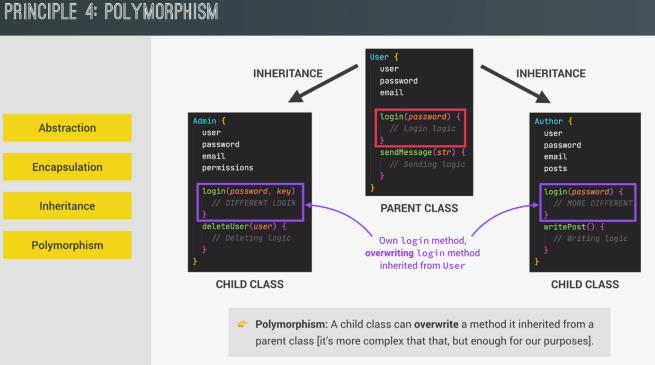
**Revealing Sections** – it’s much better if you make blur picture first before images load so it will give best performer,

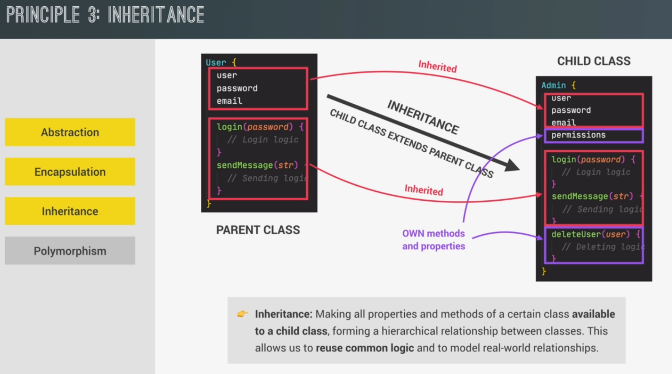
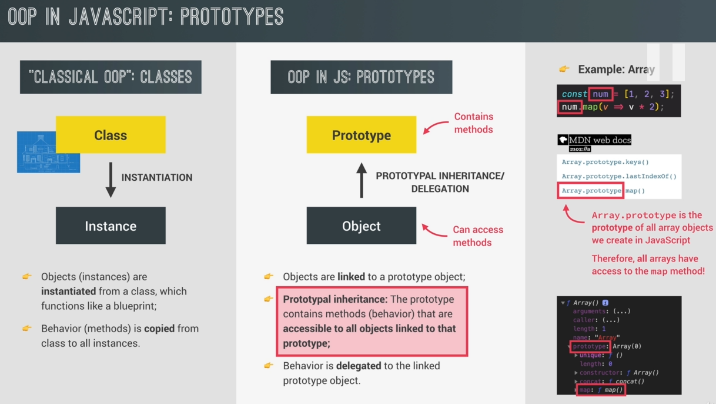
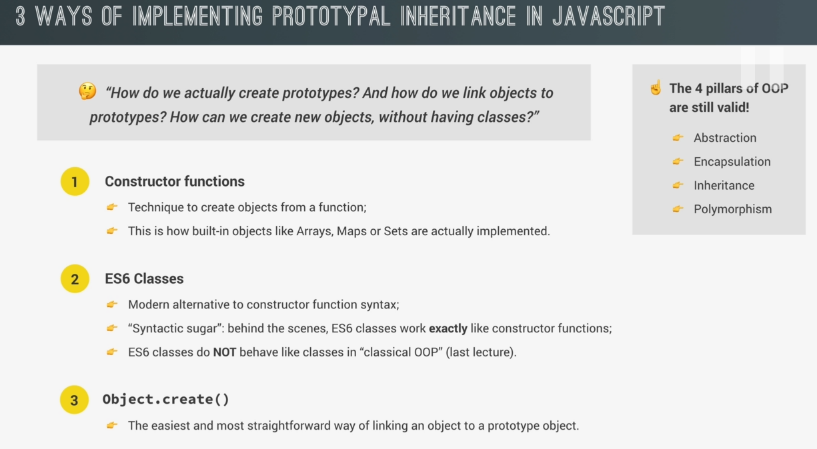
**Life cycle DOM Events** – DOMContentLoaded, load, beforeunload(don’t abuse this, this is disgusting xD)



**PART 14 Object-Oriented Programming**

 **OOP –** how we write and organize code





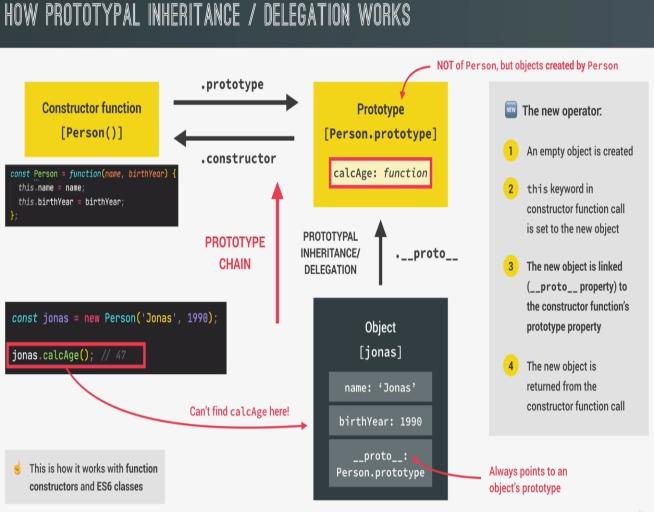
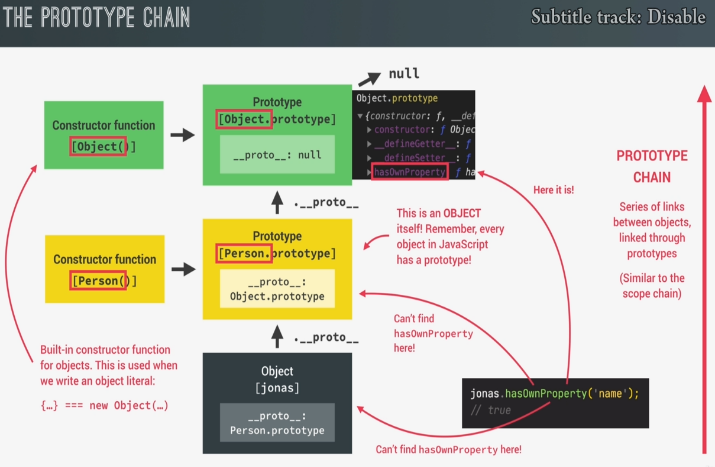
**Constructor Functions and the new Operator -**

Starts with the big letter name of function,

**Prototype –** this oop of function will make a clean version of your code. And it will improve your performance, it is like targeting but function version xD.

**Prototypal Inheritance and The Prototype Chain –** so basically there’s prototype on it because of objects, and objects has prototypes, and making this will add prototype, it’s not ending prototype

**Prototypal Inheritance on Built-In Objects –** we can do a lot of things on object, like array, function and etc. because they have prototype and it has property that we can call.

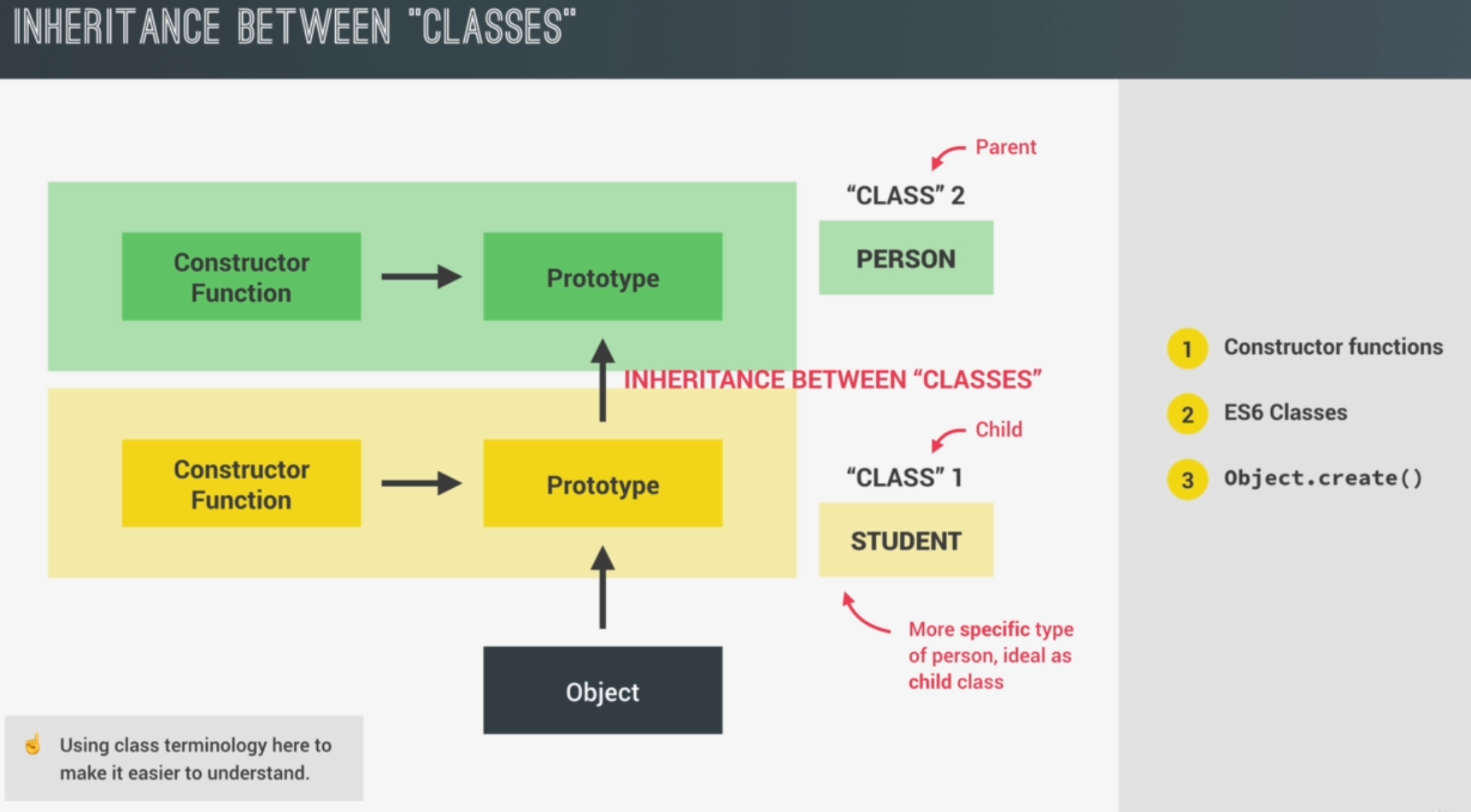


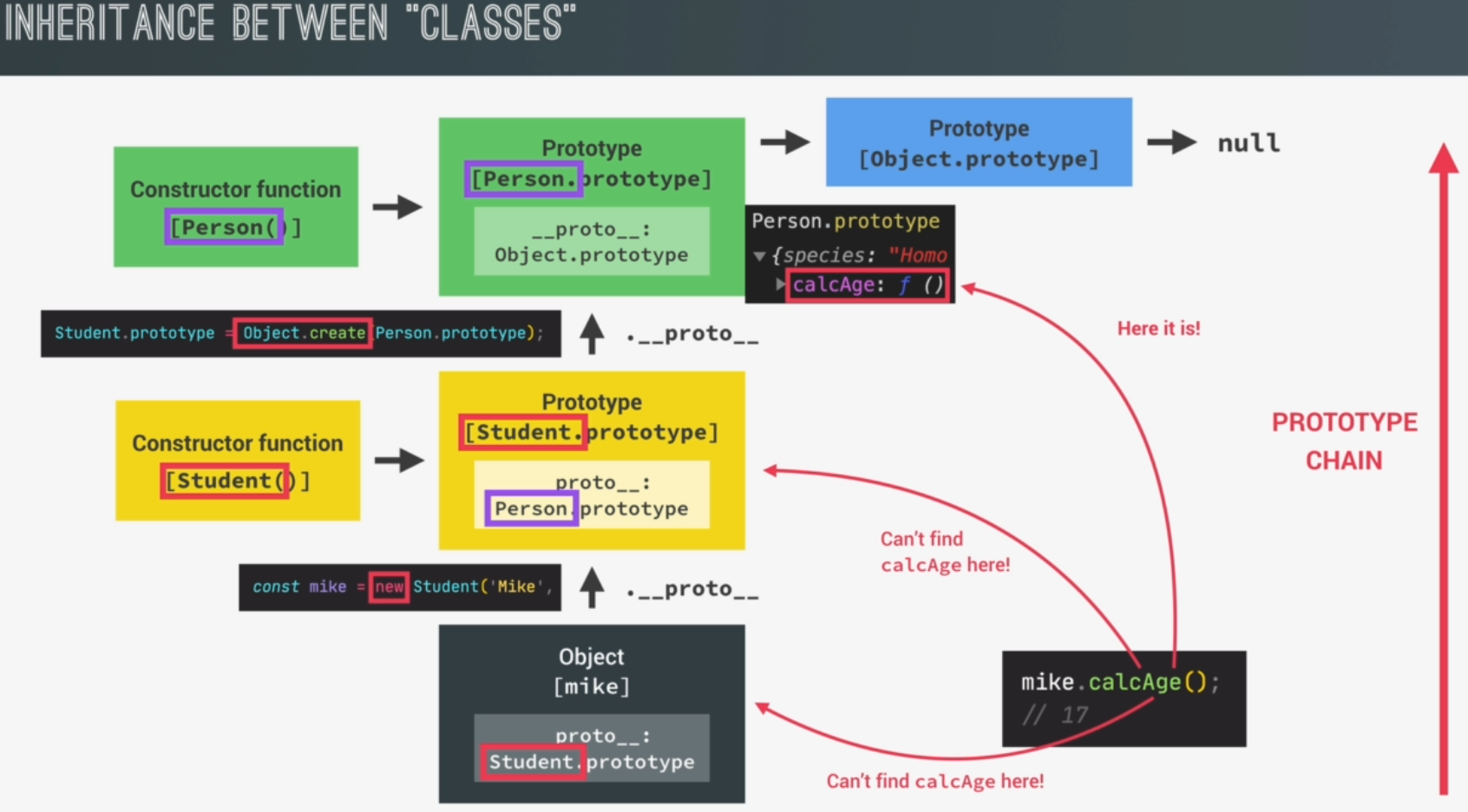
**Setters and Getters –** feature in javascript which you can use like a function

**Static Method** – will not save in prototype like setter and getters and other we use static in the first name of it.

**Object.create -**  will make an object inside of an object

**Inheritance between classes**





**Inheritance between classes ES6 Classes –** using / adding **extends** with combination of **constructor** that we’ve been using before, also using **super**

**Inheritance between classes Object.create –** use to link and easy to make a call in a property

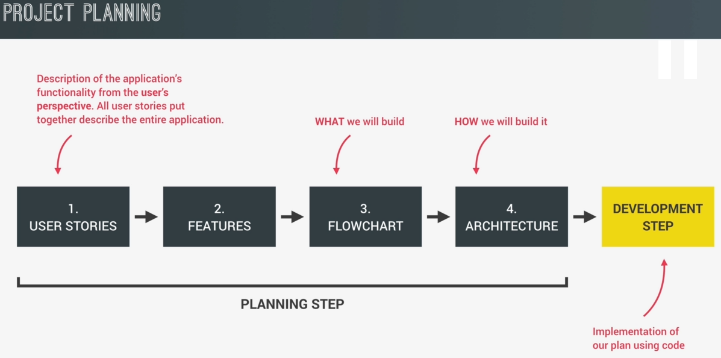
**Encapsulation Protected Properties and Methods –** probably just add underscore in the first place to secure or to remind that this data shouldn’t be use outside the object.

**Encapsulation Private Class Fields and Methods** ­ - there’s a lot of different security or private class fields, but the thing that can be access outside is using the hash # before the value or, variable,

**Chaining -** you can also use chaining when you have class

**Summary**



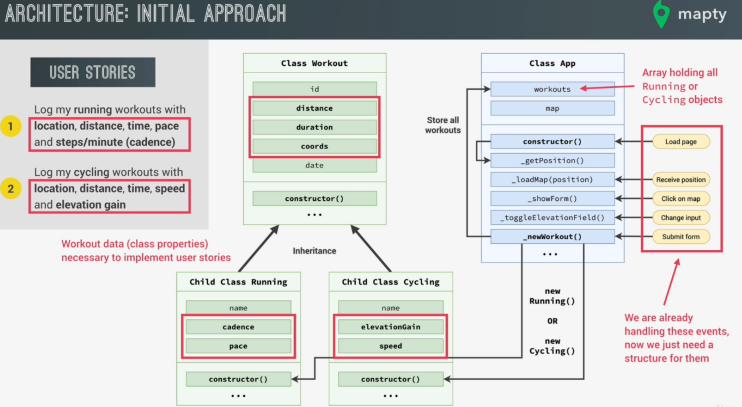
 **PART 15** **Object Oriented Programming –**

**Navigator.geolocation.getCurrentPosition –** it will show you location and will call two different function, **position.coords**

**Displaying a map –** we use a third party library by putting link in html file, and also the function of the correct execution, we could also change the theme or design of the map when changing the link inside the function.

**Third Party Libraries** – some of them have built in syntax on their on you can use you it’s pretty much important to read documentation of what you trying to use.

**Rendering Work Out –** the target + .focus(), will focus on type area, better user experience, **closest** is opposite of querySelector as it target at parents not child.

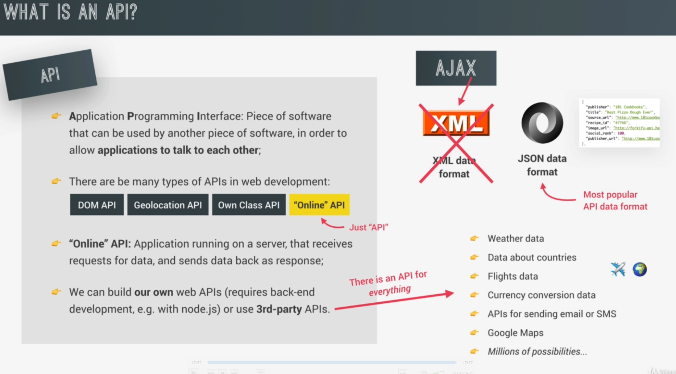
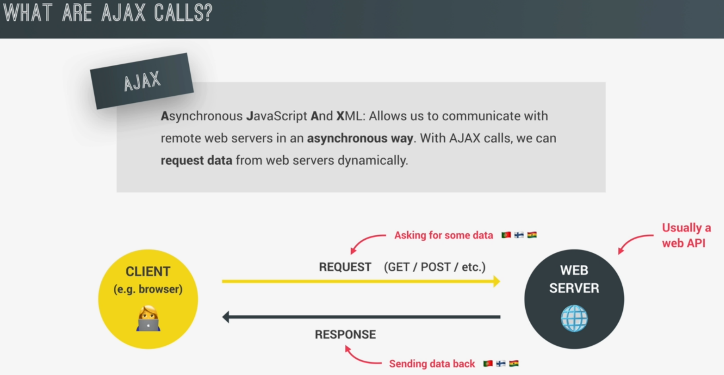
 **Project Architecture** -

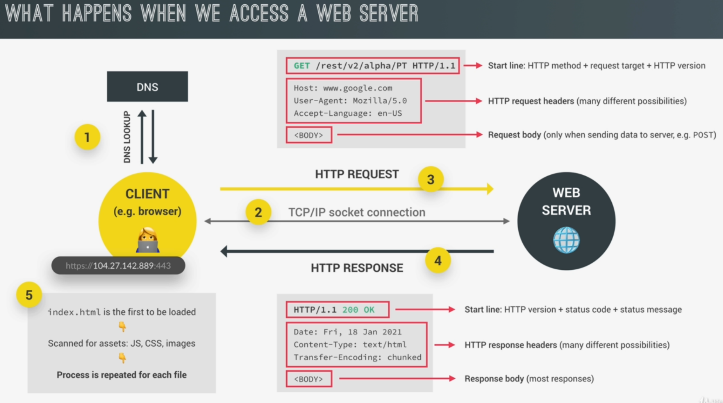
**Local storage -**  we use localStorage.setItem/getItem(function, JSON.stringify/ parse

(this.#function))

**Part 16 Asynchronous** –

Asynchronous – so it can still execute code even there is a timer

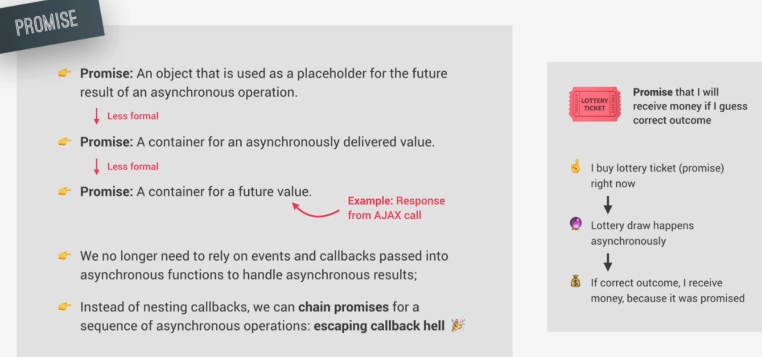


 **First Call AJAX –** new XMLHttpRequest(); open/send.(‘GET’, ‘(link)’) Rest API (should have CORS)

**Callback Hell –** sequence ofajax execution (but it makes hard to understand that it makes it bad code)

**Promises –** to avoid callback hell (using fetch) and **then**, catch to catch error inside the promises, and **finally** (will always be executed )for the end of a promises, **micro task, resolve, reject, error, catch, new Promise** (function (resolve, reject)), async, await, new Error, Promise.all(will run at the same time)

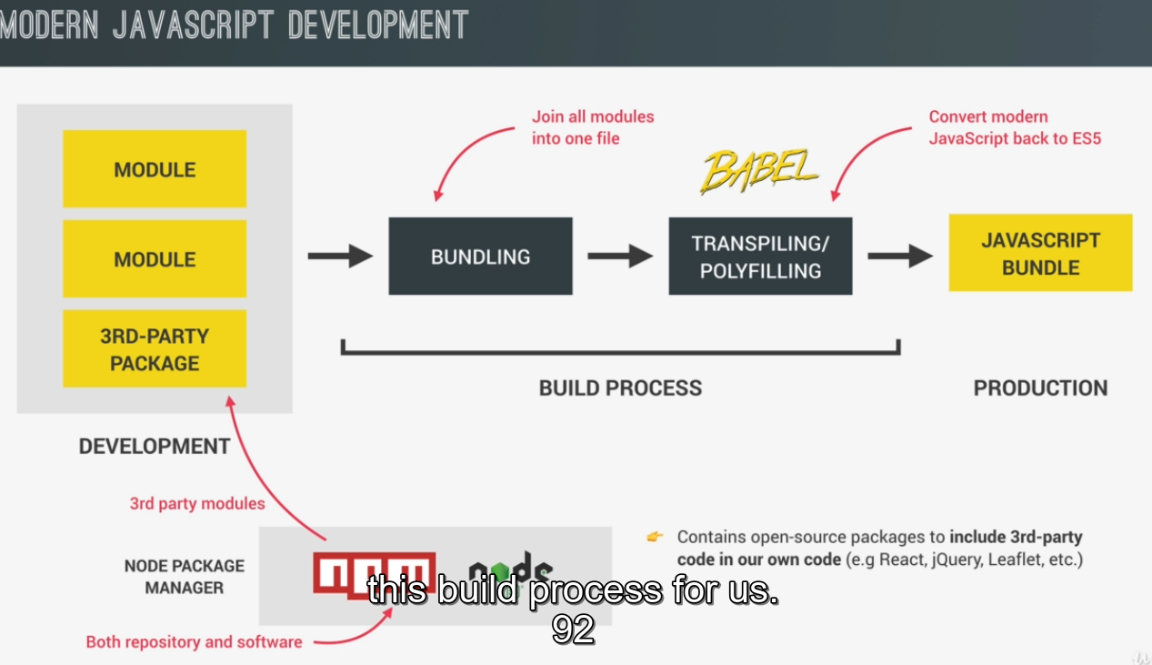
**Promise.race** (race, but rejected error)**, Promise.allSettled** (execute all)**, Promise.any** (return first fuflfill)



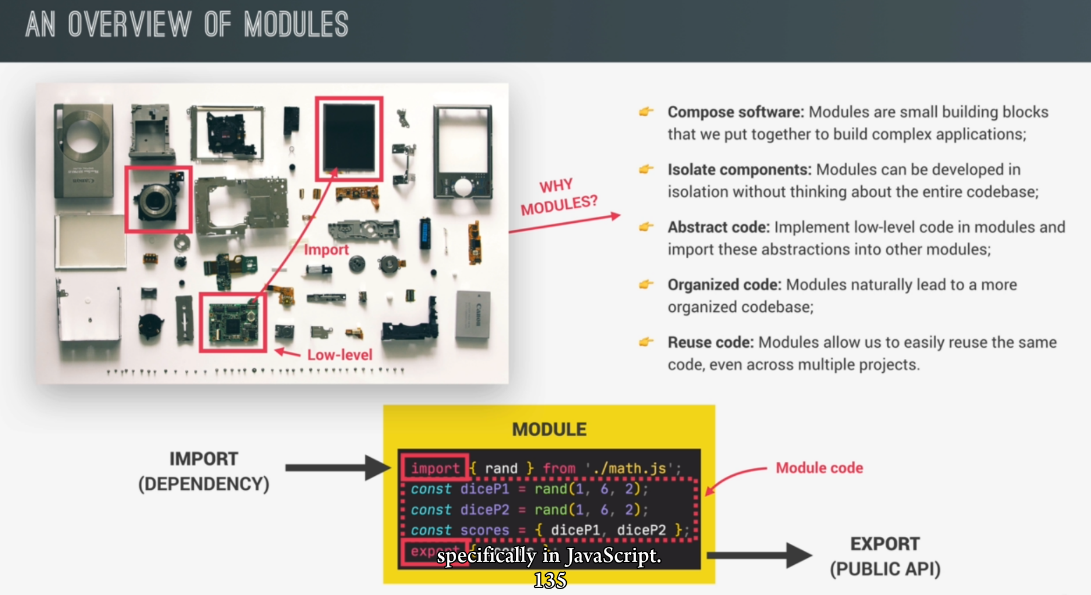
**PART 17 MODERN JAVASCRIPT**

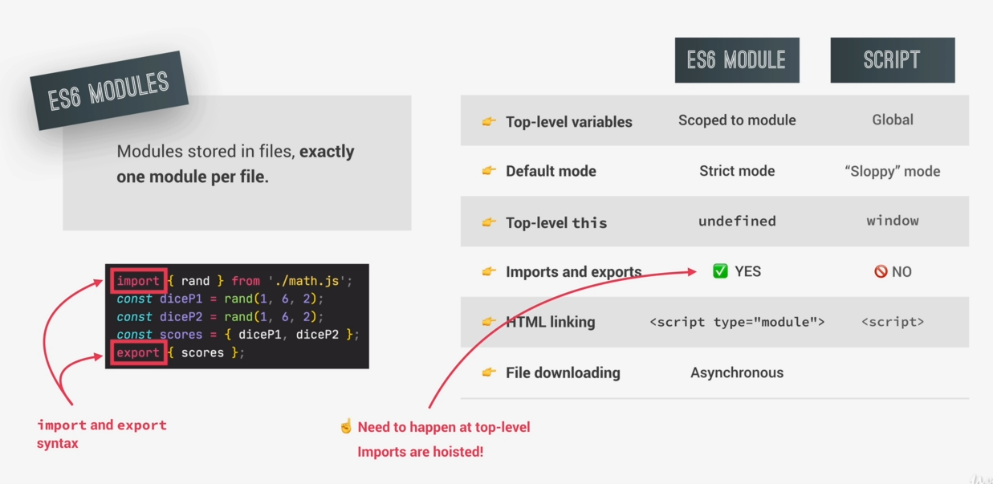
**Development** – modern javascript uses a lot of module. Even a third-party module.

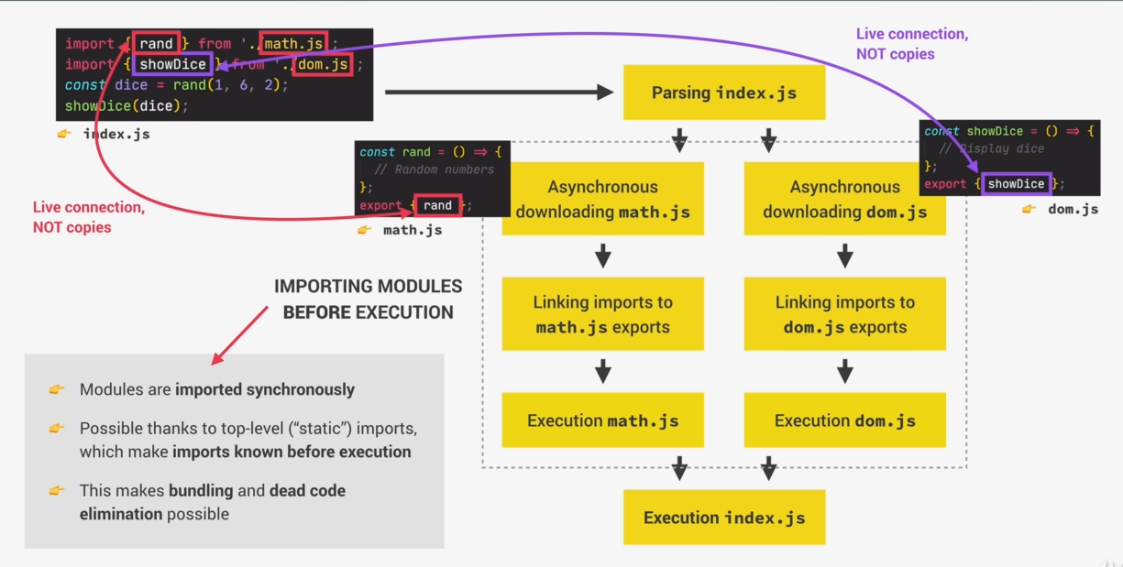
**NPM** – Node Package Development



**Webpack or Parcel** – we can use as javascript bundler

****





**Exporting Importing -** Import “./(file), on script type type=”module”, Import {(variable names)} from ‘./(file name)’ **as** change name

**Module Pattern –** by using IIFEplus returning variable we can access things inside function

**CommonJS Modules** – export.(variable) + **require**(./filelocation)

**Command Line** – to add, remove, del file.

**NPM** - Dependencies, devDependencies, bundling, pollifying, babel,

