**Array**

.slice() –kopiowanie tablicy  
.sort()  
.replace()  
.replaceAll()  
.reverse()  
.splice(który,ilelementow,..,..,..itp jakie elementy wstawiam)  
np .splice(1,1,”m”,”a”) albo .slice()1,1)/usuwa  
splice(1,0,”as”) dodanie el w dowolnym miejcu

**Array.forEach(fn(el, i\*, array\*) //nie zwraca do wyswietlania  
Array.some(fn(el, i\*, array\*) /zwraca true lub false**  
**Array.find(fn(el, i\*, array\*), this\*) zwraca el  
Array.every(fn(el, i\*, array\*)** //zwrava true albo false  
Array.map(fn(el,\*i,\*array)) //konweruje kazdy element I zwraca nowa tab  
 **Array.filter(cb)**.  
**Array.reduce(function(prev, next, i\*, arr\*) {}, start\*)**

**Konwertowanie**

parseInt(str,sys\_liczbowy)  
parseFloat(str)  
Number(str)

**Hoisting**

Unoszenie deklaracji zmiennej w przypadku var i deklaracji funkcji

**Zmienne**

Var-zasieg funkcyjny  
const,let-zasięg blokowy

**Math**

Math.PI  
.abs()  
.ceil()  
.floor()  
.max()  
.min()  
.sqrt()  
.random()\*10+1

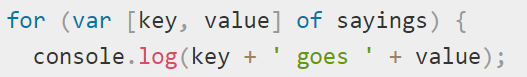
**String**  
.indexOf()  
.charAt()  
.charCodeAt()  
.toUpperCase()  
.toLowerCase()

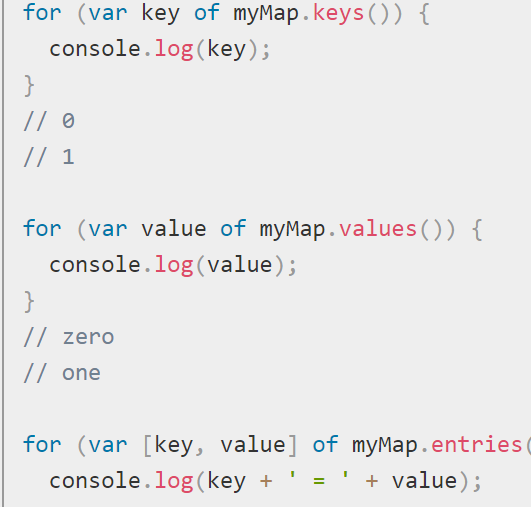
**Function**wyrażenie funkcyjne:  
const myFunction=function (){}

Deklaracja: (dostępna dla całego skryptu)  
function func(){}  
  
**funckja anonimowa**-funkcja bez nazwy  
  
Closure-funckja ktróa tworzy środowisko dla własnych zmiennych,dzieki nim mozemy miec podobny efekt do zmiennych prywatnychj

**IIFE -** **Immediately-invoked function expression**

(function() {...})(arg);

**Map**.set(key,value)  
.size  
.get()  
.has(key)  
.clear()  
.delete(key) **petla**

Object vs Map  
Map  
-brak duplikatow  
-moze zawierac obiekty ,funkcje i prymitywy,tablice

Object:  
-klucz to wart proste integer,string albo symbol

**DOM  
DOM-document object mode.**  Jest to model, interfejs, który za pomocą metod i właściwości umożliwia nam działanie na naszym dokumencie (czyli elementach naszej strony).Odwoływanie się do obiektów:

document.getElementById(‘id’)  
.. .getElementsByClassName(‘klasa’) //kolekcja (wszystko sie zmienia przy zmianach)  
.. .getElementsByTagName(‘tag’) //kolekcja(wszystko sie zmienia przy zmianach)  
.. .querySelector()  
.. . querySelectorAll()  
  
**iteracja:**

const divs = document.querySelectorAll('div.module');

Array.from(divs).forEach(div => {

console.log(div);

});

**Własciwosci elementów**.innerHTML //wnetrze obiektu (najblizsze znaczniki)  
.outerHTML //zwraca kod html obiektu

.innerText/.textContext // tekst bez tagów jak w inner html (innerText –tekst po zaaplikowaniu stylo)  
.tagName  
.setAttribute()  
.getAttribute()  
.removeAttribute()  
.hasAttribute()  
**WeakMap**klucze to obiekty  
nie ma czegos co da liste kluczy  
gdy nie ma referencji jest gotowe dla garbagecollectora  
  
**Set**-brak duplikatow  
  
.add()  
.has()  
.delte()  
.size

Set🡪Array można uzyc Array.from(mySet ) albo spreadOperatora [..mySet]   
 **WeakSet**Kolekcja obiektow , sa nie zliczalne tzn nie mozna wyswietlic ich po koeli , referencje sa trzymane slabo tzn gdy nie bd juz zadnych referencji to jest gotowe dla garabage collectora  
  
**Object**  
**this**  
Dzieki temu słowku mozna odwolac sie do obiektu z wnetrza funkcji ktora jest w nim zawarta  
-zalezy od wywoływania funkcji   
-obiekt który wywoluje funkcje staje sie this  
\*aby dostać sie do this mozna skorzystac z bind() albo funkcji strzałkowej  
  
**usuwanie właściwości**detete ten.age  
  
**\_\_proto\_\_** jest to wskaźnik na obiekt prototypu, na którym opiera się dany obiekt.  
  
**.call(this,param1,param2)** pozwala na przekazanie obiektu jako this  
 **.apply(this,[])**

**Object.create –**Tworzy obiekt który stanie się prototypem naszego obiektu  
**.hasOwnProperty(metoda/atrybut)** sprawdzamy czy obiekt ma daną metodę czy też atrybut

**Zdarzenia**

**BAZY DANYCH  
  
truncate vs delete**  
truncate usuwa całą zawartosc z tablei bez mozliwosci zrobienia rollback  
delete usuwa jeden lub wiecej rekordó  
  
**widok**to tabela do odczytu w wyniku stworzonego przez nas selecta  
  
**join rodzaje**10 rekordów tabeli select top 10  
  
**having vs where**having po grupowaniu , where przed grupowaniem having moze uzywana byc wylacznie z select  
  
**transakcja**uruchomienie seri instrukcji jako jednej. Jest atomowa  
  
**union vs union all**  
w union usuwame sa duplikaty powstałe w laczeniu zbiorw a w union all nie  
  
**funkcje agregujace**  
dla wielu wierszy zwracaja jeden wynik np. cpunt, min ,max,avg,sum  
  
**alas nadany w select nie jest widziany w where**  
where jest wykonywany po from a slect po nim   
  
**co szybsze = czy like**  
 = bo like jest skomplikowana operacja bo np %a  
  
**count(\* ) a coun(kolumna)**przy uzyciu kolumna zliczy te które nie maja nulla  
  
**bulk insertion**dodawanie danych do tabeli po jednym wierszu w celu przyspieszenia procesu  
  
**Trigger**procedura wykonywana na bazie automatycznie na zdarzenie insert,delete,upadate   
trigger nie może zatwierdzać ani anulowac transakcji  
  
**ACID**Zbiór właściwosci gwarantujacych poprawne przetwarzanie transakcji w bazach danych   
-atomicy niepodzielnosc  
-consistency spojnosc /po zatwierdzeniu transakcji maja być spełnione warunki nałożone na bazę /  
-isolation izolacja /przypadek gdy z bazy korzysta pare osob/  
 rodzaje:  
\*read uncommited  
\*read commited (domyslny)  
\* serializable  
\*repeteable reads  
\*read only  
-durability trwałość /po zakonczeniu transakcji efekty zostają w bazie/  
  
**save point**  
punkt pośredni , zapamiętany etap transakcji do którego mozna się wycofać bez anulowania wszytskich wykonywanych działań  
  
**autocommit**automatyczny zapis w bazie bo dokonaniu sql statment

**Sekwencja**  
obiekt posiadajace kolejne wartosci wg ustalonych kryteriów  
Można uzyc do generacji id   
  
**inkrementacja**Auto\_icrementent  
  
**CSS  
Selektory:**id selektory #klasa  
selektory klass .klasaelementów/tagów (div,p,body)  
atrybutów [data-modal=”open”]

,pseudo-elementów,pseudo-klas  
  
flex-dopasowanie elementów do przestrzeni ,układy na małą skalę  
grid-dwuwymiarowy system

**ES6  
let,const  
funckja strzalkowa/zmienia kontekst this na zewn/ fajne przy eventach**

**Symbol  
uproszczony zapis własciwosci w obiekcie np w wyrażeniach**const name=”kot”  
const age=32;  
const ob={ name,age,print(){console.log(this.age)}}

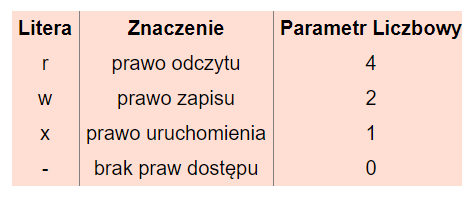
**domyslne wart**function(txt=”lorem”){  
console.log(txt)  
}

**spread syntax**const tab=[1,2,3,4]  
console.log(...tab)  
  
const tab2=[...tab]//kopia tablicy  
  
**String interpolowany**użycie ${}- mozna wywolywac metody i korzystac ze zmiennych  
 **Metody dla Stringów**startsWith(str)   
endsWith(str)  
includes(str)  
repeat(x)-zwraca nowy string bedacy powtorzeniem  
  
**Dekompozycja obiektow //podstawienie z obiektu do zmienny z tymi samymi nazwami**cons obj={ name:”kot” , surname:”kowalski”}  
const {name,surname}=obj //zmienne o takich samych nazwach  
const {name:myName,surname:mySurname}=obj  
  
**LET i CONST !**  
  
**Dekompozycja tablic**const tab=[1,2,3];  
[a,b]=tab  
  
**Object.assign(ob1,ob2)**kopiuje przeliczalne wart z ob2 do ob1 .. zawsze od prawej do lewje strony  
  
**sugar code**defniowanie klass  
  
**Promises**tworzenie obiektów które wykorzystamy do obsługi zadań asynchronicznych

**Ogólne**  
**Babel**  
zmienia nasz zapis nowy na starszy na odpowiednio transpilowalny  
  
**Webpack**  
bundler , narzędzie który czyta wskazany przez nas plik js i wykonuje na nim czynnosci. Np jesli znajdzie importowanie js to go dołączy do naszego pliku. Rozwiązuje problem dzielenia kodu na moduły. Potrafi spakować wiele róznych typów zasobów do jednego wynikowego zasobu  
  
**Gulp**odswieza okno przegladarki przy edycji,minifikuje i laczy pliki oraz style

**Grunt**  
system automatyzacji pracy.Umozliwia wykonanie taskow na danym folderze  
  
**Express**  
framework dla Node.js

**Node js**  
srodowisko uruchomieniowe dla js , jest jedno watkowe,NPM node package manager obsługuje instalację i aktualizacje modułow wielokrotnego uzytku  
**LINUX**  
**komponenty**jądro ,shell/interpreter/ ,GUI,command,script(spis komend)  
  
**bash vs dos(disc operation system)**bash ma czule na znaki komendy a dos nie  
-roznica w separatorach  
  
**CLI(command line interface)  
  
rodzaje linuxa:**fedora,slackware,debian,gentoo **rodzaje dostepu:**read,write,execute

**komendy:**  
pwd-print working directory  
ps ux –status procesu  
find .-name „os.txt” – do znalezienia  
mkdir-nowy folder  
su-zmiana użytkownika   
ps-lista procesow  
ls-zawartosc  
rmdir-usuniecie katalogu jesli pusty  
cp-copy  
rm-usuwa plik  
chmod-administrowanie uprawnieniami

x –wlasciciel  
x-grupa  
x-pozostali  
xxx  
600-prawo odczytu i zapisu dla wlasciciela  
 **Z Dupy:**  
JPA standart z grupy ORM Object Relational Model  
  
SOLID  
**Single responsibility** – klasa powinna mieć jeden powód do zmiany  
**Open/closed** –otwarty na rozszerzenia ,zamknięty na modyfikacje  
**Liskov substitution** –powinienes byc w stanie uzywac klasy pochodnej w miejscu nadrzednej  
**Interface segregation**-pilnowanie aby programista nie stworzyl systemu w którym nie mozna wprowadzac zmian  
**Dependency Inversion-**  
  
SCRUM  
  
Event loop – sprawdza czy stot jest pusty i umieszcza pierwsza rzecz z event queue  
  
Travis-integration system słuzacy do budowania i testowania kodu na githubie. Sprawdza testy przy robieniu commita

Bamboo słuzy do testowania kodu  
  
Jenkins-budowanie i testowanie oprogramowania  
  
spread syntax- zapis umożliwiający rozbijanie iterowanej wartości na składowe  
strin,tablica,kolekcje  
event delegation-  
\*nie powinno sie uzywac window.load bo czekamy tez na zaladowanie grafik  
KISS-keep it simple stupid  
DRY-Dont repeat yourself  
TDA-Tell dont ask  
  
React.js bibiblioteka  
  
Znaki ze kod jest do d:  
rigidity(sztywnosc) male zmiany prowadza do tego ze trzeba w ch przebudowac  
fragility(kruchość) zmiany w jednym module powoduja zmiany w innym  
immobility(znieruchomienie) nie da sie uzywac pownownie kodu  
viscosity(lepkosc) wprowadzenie malej zmiany to duzy koszt  
  
**Gulp**  
Narzedzie do optymalizacji pracy.Oparte na node js,uruchamia developerski serwer, odświeża okno przeglądarki