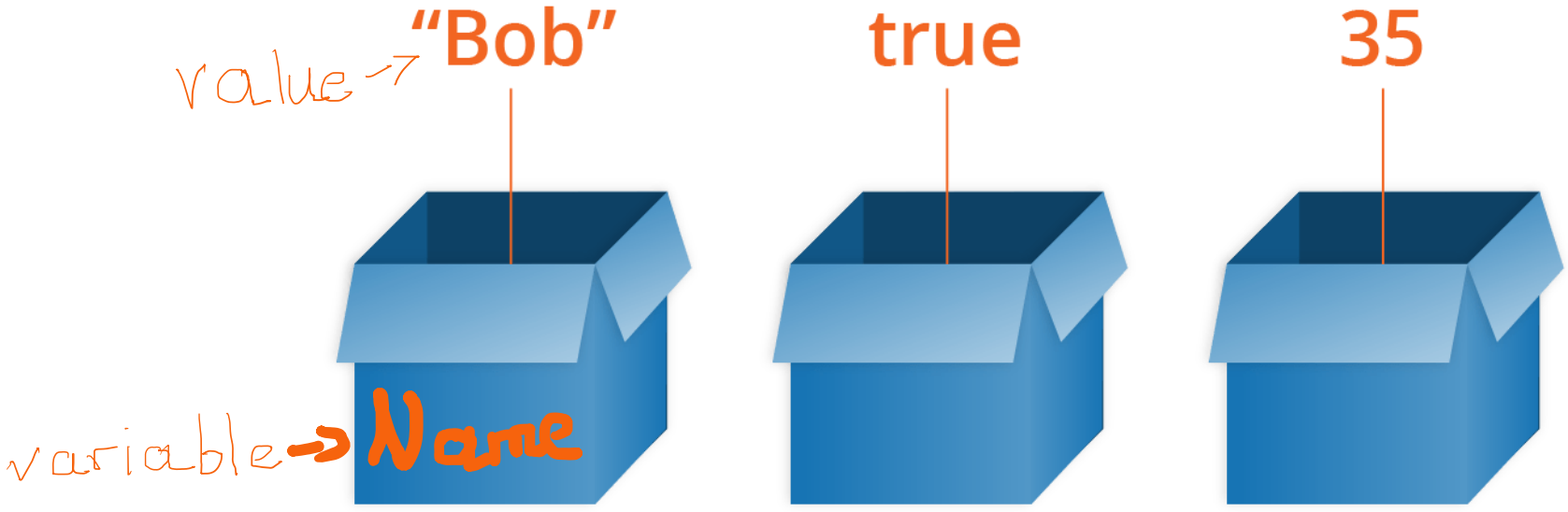
JavaScript variables

What is a variable?

- Μια μεταβλητή είναι ένα “δοχείο” για μια τιμή
    - Η τιμή αυτή θα μπορούσε να είναι ένας αριθμός που θα μπορούσαμε να χρησιμοποιήσουμε σε ένα άθροισμα
    - μια συμβολοσειρά που θα μπορούσαμε να χρησιμοποιήσουμε ως μέρος μιας πρότασης κλπ.
- Οι μεταβλητές μπορούν να περιέχουν σχεδόν οτιδήποτε — όχι μόνο συμβολοσειρές και αριθμούς.
- Οι μεταβλητές μπορούν επίσης να περιέχουν **πολύπλοκα δεδομένα** και ακόμη και ολόκληρες **συναρτήσεις**

What is a variable?

- ΠΡΟΣΟΧΗ: **Λέμε ότι οι μεταβλητές περιέχουν τιμές**.
- Αυτή είναι μια σημαντική διάκριση που πρέπει να γίνει. Οι μεταβλητές δεν είναι οι ίδιες οι τιμές, είναι δοχεία για τις τιμές. Μπορείτε να σκεφτείτε ότι είναι σαν κουτιά στα οποία μπορείτε να αποθηκεύσετε πράγματα.



JavaScript Datatypes

- Η JavaScript μας επιτρέπει να εργαζόμαστε με τους ακόλουθους τύπους δεδομένων
- Primitive values(primitive values είναι αμετάβλητες: δεν μπορούν να αλλάξουν, αν και στη μεταβλητή που την κατέχει μπορεί να εκχωρηθεί εκ νέου άλλη τιμή)
  - **Αριθμούς** είτε ακέραιους είτε δεκαδικούς χωρίς κάποια διάκριση & **Συμβολοσειρές**
  - **Boolean**: true/false.
  - **BigInt** (JavaScript BigInt variables are used to store big integer values that are too big to be represented by a normal JavaScript integer-> is only accurate up to 15 digits/ BigInt can not have decimals)
  - **Symbol** (const sym2 = Symbol("id"); Every **Symbol()** call is guaranteed to **return a unique Symbol**. Symbols are often used to add **unique property keys to an object** that won't collide with keys any other code might add to the object)
  - **Μηδέν** & **απροσδιόριστο** (null and undefined)
- complex data type: **Αντικείμενα** (objects)

JS: Untyped language

- Untyped language: μια μεταβλητή μπορεί να αποθηκεύει μια τιμή από οποιονδήποτε τύπο δεδομένων χωρίς να απαιτείται **η εκ των προτέρων δήλωση του τύπου**

- Ο τύπος της τιμής που αποθηκεύεται μπορεί να αλλάξει κατά την εκτέλεση του κώδικα και η JavaScript το διαχειρίζεται κατάλληλα

Μεταβλητές στη JavaScript

- Υπάρχουν 3ς διαφορετικές λέξεις κλειδιά για τον ορισμό μεταβλητών:

- var
- let
- const

Var

- JavaScript variables: μπορούν να θεωρηθούν ως δοχεία για τα δεδομένα
- var: declaration of variables

```
<script>
    var name;
    var surname;
</script>
```

var

```
<script>
    var name;
</script>
```

- η μεταβλητή δεν έχει ακόμη τιμή.
- Η προεπιλεγμένη τιμή των μεταβλητών που δεν έχουν καμία τιμή είναι **undefined**.

var

- Μπορούμε να **εκχωρήσουμε μια τιμή** σε μια μεταβλητή χρησιμοποιώντας τον τελεστή =
  a. όταν την δηλώσουμε
  b. ή μετά τη δήλωση και πριν την πρόσβαση σε αυτήν

```
var name= “Aristea”;
var surname;
surname=“Kontogianni”;
```

Πολλές μεταβλητές μπορούν επίσης να δηλωθούν ταυτόχρονα σε μία μόνο γραμμή

```
var name= “Aristea”, surname;
```

Ας δούμε ένα παράδειγμα…

let

Λέξη – κλειδί let : υπάρχουν οι ορισμένες διαφορές σε σχέση με τη λέξη – κλειδί var. Οι μεταβλητές που ορίζονται με let:

- δεν μπορούν να δηλωθούν ξανά
- πρέπει να δηλώνονται πριν από τη χρήση
- έχουν Block Scope : η περιοχή μέσα σε ένα if, switch , for και while . Γενικότερα όποτε βλέπουμε {curly brackets} είναι ένα block

let

Οι μεταβλητές που ορίζονται με let **δεν μπορούν να δηλωθούν ξανά**

```
let name = "Aristea Kontogianni";
let name = 0; //Uncaught SyntaxError: Identifier ‘name' has already been declared
```

Or

```
var name = 0; //Uncaught SyntaxError: Identifier ‘name’ has already been declared
```

let

Οι μεταβλητές που ορίζονται με let πρέπει να δηλώνονται πριν από τη χρήση

```
myName = "Aristea";
let myName = "Aristea Kontogianni";
```

```
▶Uncaught ReferenceError: Cannot access 'myName'
before initialization
```

let

Οι μεταβλητές που ορίζονται με let/const έχουν Block Scope

- Block scope (w3schools): “Variables declared inside a { } block cannot be accessed from outside the block”

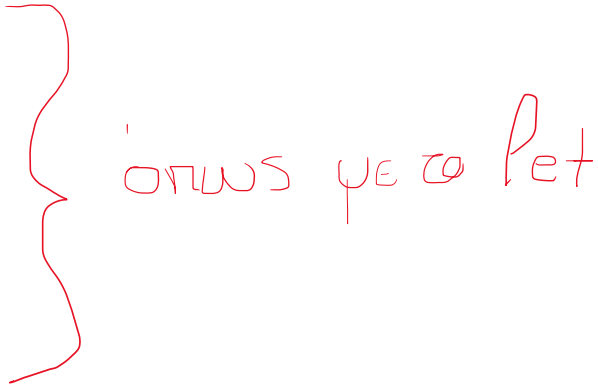
```
function foo(){
if(true){
    var fruit1 = 'apple';         //exist in function scope
     const fruit2 = 'banana';      //exist in block scope
    let fruit3 = 'strawberry';    //exist in block scope

}
console.log(fruit1); //apple
console.log(fruit2); //Uncaught ReferenceError: fruit2 is not defined at foo
console.log(fruit3); //Uncaught ReferenceError: fruit3 is not defined at foo
```

let example

```
let x = 1;
if (x === 1) {
  let x = 2;
  console.log(x);
  document.getElementById("block").innerHTML = x ;
  // expected output: 2
}
document.getElementById("demo").innerHTML = x ;

console.log(x);
// expected output: 1
```

Const

- Οι μεταβλητές που ορίζονται με τη λέξη – κλειδί const
- δεν μπορούν να δηλωθούν ξανά
- πρέπει να δηλώνονται πριν από τη χρήση
- έχουν Block Scope

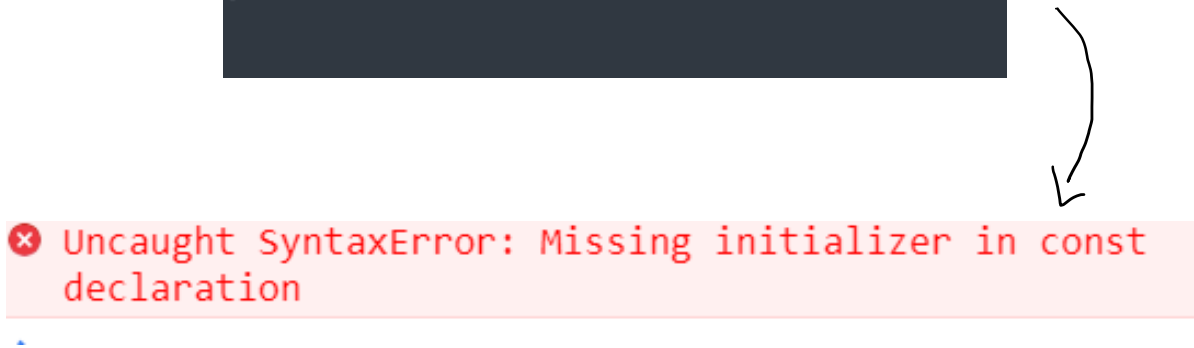


- πρέπει αρχικοποιούνται όταν δηλώνονται
- Επίσης, δεν μπορούμε να εκχωρήσουμε εκ νέου τιμή σε const variable

Const

- Οι μεταβλητές που ορίζονται με τη λέξη – κλειδί const πρέπει αρχικοποιούνται όταν δηλώνονται

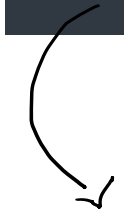
```
const PI;
PI = 3.14;
```



Const

- Δεν μπορούμε να εκχωρήσουμε εκ νέου τιμή σε const variable

```
const PI = 3.141592653589793;
PI = 3.14;
PI = PI + 10;
```



```
▶ Uncaught TypeError: Assignment to constant variable.
```

Const

- Χρησιμοποιώντας τη λέξη – κλειδί const δημιουργούμε μια read-only αναφορά σε μια τιμή

- Αυτό **δεν σημαίνει ότι η τιμή που διατηρεί είναι αμετάβλητη** - απλώς ότι **το όνομα της μεταβλητής δεν μπορεί να εκχωρηθεί εκ νέου**.

- Για παράδειγμα, στην περίπτωση που το περιεχόμενο μιας const μεταβλητής είναι ένα **αντικείμενο**, τα περιεχόμενά του (π.χ., οι ιδιότητές του) μπορούν να τροποποιηθούν.

Παράδειγμα

- Η JavaScript υποστηρίζει την δημιουργία objects, δηλαδή αντικειμένων
- Τα αντικείμενα είναι επίσης μεταβλητές. Αλλά τα αντικείμενα μπορούν να περιέχουν πολλές τιμές.
- const person = {
  firstName: “Aristea",
  lastName: “Kontogianni",
  age: 19,
  eyeColor: "blue"
  };

Const

Έστω ότι έχουμε το παρακάτω array, τι περιμένουμε να συμβεί?

```
const cats = ["javie", "rosa", "gatoulis"];

cats[0] = "Javie";

cats.push("miaou");

console.log(cats);
```

Const

```
// You can create a constant array:
const cats = ["javie", "rosa", "gatoulis"];

// You can change an element:
cats[0] = "Javie";

// You can add an element:
cats.push("miaou");

console.log(cats);
```

▶ (4) ['Javie', 'rosa', 'gatoulis', 'miaou']

Const

Το παρακάτω θα δουλέψει?

```
const cats = ["javie", "rosa", "gatoulis"];

cats = ["Javie", "rosa", "gatoulis","miaou"];
```

Const

Δεν μπορούμε να εκχωρήσουμε εκ νέου τιμή στο cats

```
const cats = ["javie", "rosa", "gatoulis"];

cats = ["Javie", "rosa", "gatoulis","miaou"];
```

```
Uncaught TypeError: Assignment to constant variable.
```

Ερώτηση!

- Ποιες από τις παρακάτω γραμμές κώδικα επιτρέπονται?

```
 6   const x = 2;
 7   const x = 3;
 8   x = 3;
 9   var y = 3;
10   let y = 3;
11
12   {
13     const x = 3;
14     x = 3;
15     let y = 3;
16   }
17
```

Ερώτηση!

- Ποιες από τις παρακάτω γραμμές κώδικα επιτρέπονται?

```
6    const x = 2;
7    const x = 3;
8    x = 3;
9    var y = 3;
10   let y = 3;
11
12   {
13     const x = 3;
14     x = 3;
15     let y = 3;
16   }
17
```

```
<!DOCTYPE html>
<html>
<body>
<script>

const x = 2;        // Allowed
//const x = 3;        // Not allowed
//x = 3;              // Not allowed
var y = 3;          //  allowed
//let y = 3;          // Not allowed

{
  const x = 3;    // Allowed
  //x = 3;          // Not allowed
  let y = 3;      // allowed
}

</script>
```

JavaScript Identifiers

- JavaScript variables : ορίζονται χρησιμοποιώντας κάθε φορά μοναδικά ονόματα
- unique names : identifiers

Δημιουργία ονομάτων μεταβλητών:

- Αποδεκτοί χαρακτήρες: γράμματα, ψηφία, underscores, και dollar signs.
- Τα ονόματα των μεταβλητών μπορούν να ξεκινήσουν με ένα γράμμα ή με $ και _

JavaScript Identifiers

- Δημιουργία ονομάτων μεταβλητών:
- Τα ονόματα είναι case sensitive
  - var name, Name //different variables

- Οι δεσμευμένες λέξεις δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών
- i.e. for, int, static, true etc

Variable Scope

- Στην JavaScript οι μεταβλητές μπορεί να είναι
  - **Global Variables:** είναι προσβάσιμες από παντού στον κώδικα (ακόμα και από μέσα από τις συναρτήσεις) και δηλώνονται έξω από οποιαδήποτε συνάρτηση
  - **Local Variables** : Είναι ορατή μόνο μέσα στην συνάρτηση στην οποία ορίζεται

Variable Scope

- Οι μεταβλητές μπορούν να δηλωθούν και να αρχικοποιηθούν χωρίς τη λέξη -κλειδί var
- Οι μεταβλητές που δηλώνονται χωρίς τη λέξη -**κλειδί var είναι global μεταβλητές**, ανεξάρτητα από το πού δηλώνονται
- Για τον λόγο αυτό δεν συνιστάται η δήλωση μιας μεταβλητής χωρίς τη λέξη -κλειδί var - > μπορεί κατά λάθος να αντικαταστήσει μια υπάρχουσα global μεταβλητή

Ερώτηση

- Τι θα προβληθεί στα στοιχεία με id=demo και id=exp ;

```
<p id="demo"></p>
<p id="exp"></p>

<script>
    var name = "Aristea";
    var surname = " Kontogianni";
    var z = name + surname;
    var x=4, y=3, total=x+y;

    document.getElementById("demo").innerHTML = z ;
    document.getElementById("exp").innerHTML = "<span style=\"font-size:40px;\">"+total+"
        </span>";
</script>
```

Απάντηση

- Τι θα προβληθεί στα στοιχεία με id=demo και id=exp ;

```
<p id="demo"></p>
<p id="exp"></p>

<script>
    var name = "Aristea";
    var surname = " Kontogianni";
    var z = name + surname;
    var x=4, y=3, total=x+y;

    document.getElementById("demo").innerHTML = z ;
    document.getElementById("exp").innerHTML = "<span style=\"font-size:40px;\">"+total+"
        </span>";
</script>
```

Aristea Kontogianni

7

Συμβολοσειρές (Strings)

- Οι συμβολοσειρές στη JavaScript, χρησιμοποιούνται για την αποθήκευση και την επεξεργασία κειμένου

- Μια συμβολοσειρά αποτελείται από έναν αριθμό χαρακτήρων μέσα σε εισαγωγικά (μονά/διπλά)

- var name = “Aristea Kontogianni";

Συμβολοσειρές (Strings)

- Μέσα σε ένα string μπορούν να υπάρχουν επιπλέον εισαγωγικά αρκεί να διαφέρουν από αυτά που το ορίζουν:

var details= “You can call me ‘Aristea’ ”

Συμβολοσειρές (Strings)

- Ορισμένες από τις ενέργειες που κάνουμε με τις συμβολοσειρές είναι:
- Να βρίσκουμε το μήκος τους : string.length
- Να ενώσουμε 2 συμβολοσειρές:
  - Var str1 = 'Hello ‘ + “ ”+ "World " ;
  - var str2 = " Everyone!";
  - **var res = str1.concat(str2);**
  - Τι string περιέχει το res?

Συμβολοσειρές (Strings)

- Ορισμένες από τις ενέργειες που κάνουμε με τις συμβολοσειρές είναι:
- Να βρίσκουμε το μήκος τους : string.length
- Να ενώσουμε 2 συμβολοσειρές:
  - Var str1 = 'Hello ' + "World " ;
  - var str2 = " Everyone!";
  - var res = str1.concat(str2);

  - Hello World Everyone!

Συμβολοσειρές (Strings)

- Ας δούμε και το res.length…

Συμβολοσειρές (Strings)

- Ορισμένες από τις ενέργειες που κάνουμε με τις συμβολοσειρές είναι:

- Να ελέγχουμε για την ύπαρξη ή τη θέση ενός substring με τη μέθοδο **indexOf () //-1** αν δεν υπάρχει

- **const paragraph = 'The lazy dog’;**
- **const searchTerm = 'dog’;**
- **const indexOfFirst = paragraph.indexOf(searchTerm);**
- **//The index of "dog" is 9'**

Συμβολοσειρές (Strings)

εξαγάγουμε substrings με τη μέθοδο :substring()

- var res = str.substring(1, 5);

- Η μέθοδος substring () εξάγει χαρακτήρες, μεταξύ των δεικτών (1,5 εδώ), από μια συμβολοσειρά και επιστρέφει την υπο –συμβολοσειρά
- //προσέξτε τον δεύτερο δείκτη δεν τον περιλαμβάνει στους χαρακτήρες

Συμβολοσειρές (Strings)

Τι περιμένουμε να δούμε στην κονσόλα στο παρακάτω παράδειγμα?

```
var str = "Have a nice day!";
 var res = str.substring(1, 5);
 console.log(res);
 console.log(str);
```

Συμβολοσειρές (Strings)

Τι περιμένουμε να δούμε στην κονσόλα στο παρακάτω παράδειγμα;

```
var str = "Have a nice day!";
var res = str.substring(1, 5);
console.log(res);
console.log(str);
```

```
ave
Have a nice day!
```

Συμβολοσειρές Μέθοδοι

charAt(position)	Returns the character at the specified position (in Number).
charCodeAt(position)	Returns a number indicating the Unicode value of the character at the given position (in Number).
concat([string,,])	Joins specified string literal values (specify multiple strings separated by comma) and returns a new string.
indexOf(SearchString, Position)	Returns the index of first occurrence of specified String starting from specified number index. Returns -1 if not found.
lastIndexOf(SearchString, Position)	Returns the last occurrence index of specified SearchString, starting from specified position. Returns -1 if not found.
localeCompare(string,position)	Compares two strings in the current locale.
match(RegExp)	Search a string for a match using specified regular expression. Returns a matching array.
replace(searchValue, replaceValue)	Search specified string value and replace with specified replace Value string and return new string. Regular expression can also be used as searchValue.
search(RegExp)	Search for a match based on specified regular expression.

slice(startNumber, endNumber)	Extracts a section of a string based on specified starting and ending index and returns a new string.
split(separatorString, limitNumber)	Splits a String into an array of strings by separating the string into substrings based on specified separator. Regular expression can also be used as separator.
substr(start, length)	Returns the characters in a string from specified starting position through the specified number of characters (length).
substring(start, end)	Returns the characters in a string between start and end indexes.
toLocaleLowerCase()	Converts a string to lower case according to current locale.
toLocaleUpperCase()	Converts a sting to upper case according to current locale.
toLowerCase()	Returns lower case string value.
toString()	Returns the value of String object.
toUpperCase()	Returns upper case string value.
valueOf()	Returns the primitive value of the specified string object.

Table source: https://www.tutorialsteacher.com/

Strings είναι immutable

Οι συμβολοσειρές στη JavaScript είναι αμετάβλητες

- δεν μπορούν να τροποποιηθούν μόλις δημιουργηθούν.
- μπορείτε να δημιουργήσετε μια νέα συμβολοσειρά από μια υπάρχουσα συμβολοσειρά.

Strings είναι immutable

```
var test = 'JavaScript';
test = test + 'Something';
console.log(test);


test[0] = 'X';
console.log(test)
console.log(test[0]);
```

Objects

- Η JavaScript υποστηρίζει την δημιουργία objects, δηλαδή αντικειμένων
- Ένα αντικείμενο στη JavaScript είναι μια **συλλογή ονοματισμένων τιμών**
- Στα αντικείμενα λοιπόν έχουμε ζευγάρια key-value (name-value)
- Τα αντικείμενα ορίζονται με τις λέξεις κλειδιά var-let-const

Objects

```
const cat = {
  name: "Javie",
  age: "5",
  color: "white",
  eyeColor: "blue"
};

console.log(cat["name"]);
console.log(cat.age);
```

Objects

- Για να αποκτήσουμε πρόσβαση σε μια ιδιότητα ενός αντικειμένου, χρησιμοποιούμε:

- dot (.)
  - objectName.propertyName

- Array-like notation ([])
  - objectName['propertyName’]

- objectName[expression] // x = " 'propertyName "; objectName[x]

- Αν προσπαθήσουμε να διαβάσουμε μια ιδιότητα που δεν υπάρχει: undefined
- Μπορούμε να διαγράψουμε μια ιδιότητα μέσω του delete operator **//delete cat.name;**
- in operator: υπάρχει μια ιδιότητα σε ένα αντικείμενο?

Objects

- Επιπλέον μπορούν να χρησιμοποιηθούν μεταβλητές ως ιδιότητες

```
var name="Aristea";
var surname = " Kontogianni";
var person = { name, surname };
console.log(person);
```

Objects

- Επιπλέον μπορούν να χρησιμοποιηθούν μεταβλητές ως ιδιότητες

```
var name="Aristea";
var surname = " Kontogianni";
var person = { name, surname };
console.log(person);
```

Objects

- Τρόποι για να αποκτήσουμε πρόσβαση στις ιδιότητες ενός αντικειμένου :

```
var name="Aristea";
var surname = " Kontogianni";
var person = { name, surname };
console.log(person.name+" "+person["surname"]);
```

```
Aristea  Kontogianni
```

Objects

- Επίσης μπορούμε να προσθέσουμε εμφωλευμένο αντικείμενο μέσα σε ένα object
- Κάθε αντικείμενο στη JavaScript μπορεί να μετατραπεί σε συμβολοσειρά με τη συνάρτηση JSON.stringify ():

```
var name="Aristea";
var surname = " Kontogianni";
var person = { name, surname };
console.log(person.name+" "+person["surname"]);

var city ="Athens";
var country="Greece";
person.address={city,country};
console.log(person);
document.body.innerHTML = JSON.stringify(person);
```

{"name":"Aristea","surname":" Kontogianni","address":{"city":"Athens","country":"Greece"}}

Objects

- JSON.parse()->string becomes a JavaScript object

```
var str= '{"name":"John", "age":30, "city":"New York"}';
console.log(typeof(str));
console.log(str);

const obj = JSON.parse(str);
console.log(typeof(obj));
console.log(obj);
```

Objects

- Η μέθοδος Object.freeze() παγώνει ένα αντικείμενο.
- Το πάγωμα ενός αντικειμένου καθιστά τις υπάρχουσες ιδιότητες **μη εγγράψιμες** και **μη διαμορφώσιμες**.
- Δεν είναι πλέον δυνατή η αλλαγή ενός παγωμένου αντικειμένου: δεν μπορούν να προστεθούν νέες ιδιότητες, δεν είναι δυνατή η κατάργηση υπαρχουσών ιδιοτήτων, ούτε να αλλάξουν.
- Η freeze() επιστρέφει το ίδιο αντικείμενο
- Note that: η μέθοδος αυτή δεν “Παγώνει” nested objects/arrays.

More on Objects

- let user = {

  name: “aristea”,

  surname: “Kontogianni”,

  age: “23”

  }; // "object literal" syntax

Άλλος τρόπος δημιουργίας object : **new** keyword

```
let person = new Object();
person.name="John";
person.surname="Black";
person.age=30;
```

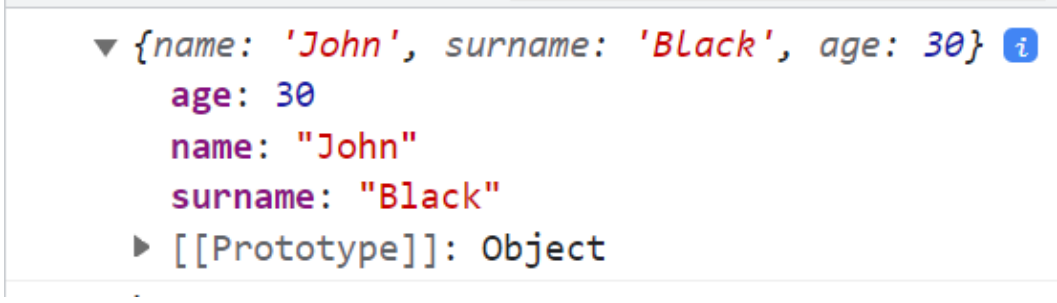
Prototype inheritance

- Κάθε object στη JavaScript έχει μια ενσωματωμένη ιδιότητα, η οποία ονομάζεται **πρωτότυπο** (prototype).
- Το πρωτότυπο είναι από μόνο του ένα **αντικείμενο**, επομένως το **πρωτότυπο** θα έχει το δικό του πρωτότυπο, δημιουργώντας αυτό που ονομάζεται prototype chain.
- Η αλυσίδα τελειώνει όταν φτάσουμε σε ένα πρωτότυπο που έχει null για το δικό του πρωτότυπο.
- Τα Prototypes είναι ο μηχανισμός με τον οποίο τα JavaScript objects κληρονομούν χαρακτηριστικά το ένα από το άλλο

Prototype inheritance

- Τα Prototypes είναι ο μηχανισμός με τον οποίο τα JavaScript objects κληρονομούν χαρακτηριστικά το ένα από το άλλο

- Όλα τα JavaScript objects κληρονομούν ιδιότητες και μεθόδους από ένα πρωτότυπο (Prototype), το οποίο είναι ένα object: Object.prototype



Prototype inheritance

- Όταν προσπαθούμε να αποκτήσουμε πρόσβαση σε μια ιδιότητα ενός αντικειμένου: εάν η ιδιότητα δεν μπορεί να βρεθεί στο ίδιο το αντικείμενο, γίνεται αναζήτηση στο prototype για την ιδιότητα.

- Εάν η ιδιότητα εξακολουθεί να μην μπορεί να βρεθεί, τότε γίνεται αναζήτηση του πρωτοτύπου του πρωτοτύπου και ούτω καθεξής, μέχρι να φτάσουμε στο τέλος της αλυσίδας και να πάρουμε **undefined**

Object create

- Η μέθοδος Object.create() **δημιουργεί ένα νέο αντικείμενο**, χρησιμοποιώντας ένα **υπάρχον αντικείμενο ως πρωτότυπο** του νεοδημιουργημένου αντικειμένου.

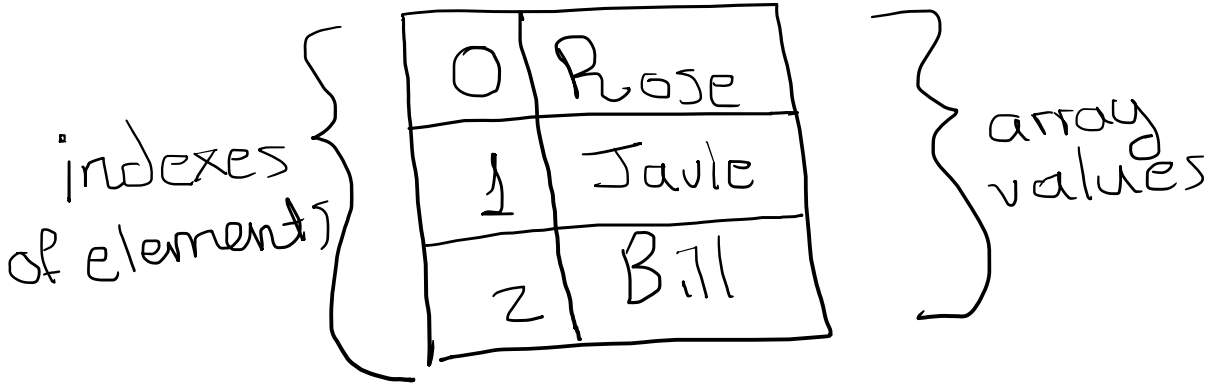
```
const person = {
  name: 'Aristea',
  age: 31
};

var me = Object.create(person);

me.surname = 'Kontogianni'; // "sur
me.email = 'test@gmail.com'; // "em

console.log(me);
console.log(me.name);
console.log(person.surname);
```

```
object_create.html:21
▼ {surname: 'Kontogianni', email: 'test@gmail.com'} i
    email: "test@gmail.com"
    surname: "Kontogianni"
  ▼ [[Prototype]]: Object
      age: 31
      name: "Aristea"
    ▼ [[Prototype]]: Object
      ▶ constructor: ƒ Object()
      ▶ hasOwnProperty: ƒ hasOwnProperty()
      ▶ isPrototypeOf: ƒ isPrototypeOf()
      ▶ propertyIsEnumerable: ƒ propertyIsEnumerable()
      ▶ toLocaleString: ƒ toLocaleString()
      ▶ toString: ƒ toString()
      ▶ valueOf: ƒ valueOf()
      ▶ __defineGetter__: ƒ __defineGetter__()
      ▶ __defineSetter__: ƒ __defineSetter__()
      ▶ __lookupGetter__: ƒ __lookupGetter__()
      ▶ __lookupSetter__: ƒ __lookupSetter__()
        __proto__: (...)
      ▶ get __proto__: ƒ __proto__()
      ▶ set __proto__: ƒ __proto__()
Aristea                                  object_create.html:22
undefined                                object_create.html:23
>
```


```
▶ {surname: 'Kontogianni', email: 'test@gmail.com'}  object_create.html:21
Aristea                                              object_create.html:22
undefined                                            object_create.html:23
>
```

Arrays

- Ένα array (πίνακας) αποτελεί μια μεταβλητή που περιέχει περισσότερες από μία τιμές κάθε φορά
- Στη μνήμη, ένα array αντιπροσωπεύει μια ομάδα τοποθεσιών μνήμης
- Κάθε μεμονωμένη τοποθεσία ονομάζεται στοιχείο (element)



Arrays

- Cats -> Δημιουργούμε ένα array με τέσσερα στοιχεία, τα οποία ορίζονται κατά τη δήλωση του πίνακα

```
var cats = ["Ragdoll", "Bengal", "Ocicat","Toyger"];
document.getElementById("demo").innerHTML = cats;
```

Arrays

- Πόσα στοιχεία περιέχουν οι παρακάτω πίνακες;

```
<script>
var cats = ["Ragdoll", "Bengal", "Ocicat","Toyger"];
var cats1 = ["Ragdoll",,,,"Toyger"];
```

Arrays

- Cats -> Δημιουργούμε ένα array με **τέσσερα** στοιχεία, τα οποία **ορίζονται κατά τη δήλωση** του πίνακα

- Cats1-> Δημιουργεί ένα array με **5** στοιχεία, **3** από τα οποία δεν έχουν καθοριστεί ακόμη

```
var cats1 = ["Ragdoll", , , ,"Toyger"];
cats1[1]="Bengal";
document.body.innerHTML =cats1;
```

Ragdoll,Bengal,,,Toyger

typeof Array

- Οι πίνακες είναι στην πραγματικότητα ένας ειδικός τύπος αντικειμένων
- Έτσι, typeof Array επιστρέφει “object"
- Ωστόσο, τα arrays στη JavaScript είναι καλύτερο να περιγράφονται ως arrays

Arrays' Elements

Ένα array μπορεί να περιέχει :

- μεταβλητές διαφορετικών τύπων
- αντικείμενα
- συναρτήσεις
- άλλα arrays

Stacks/queues

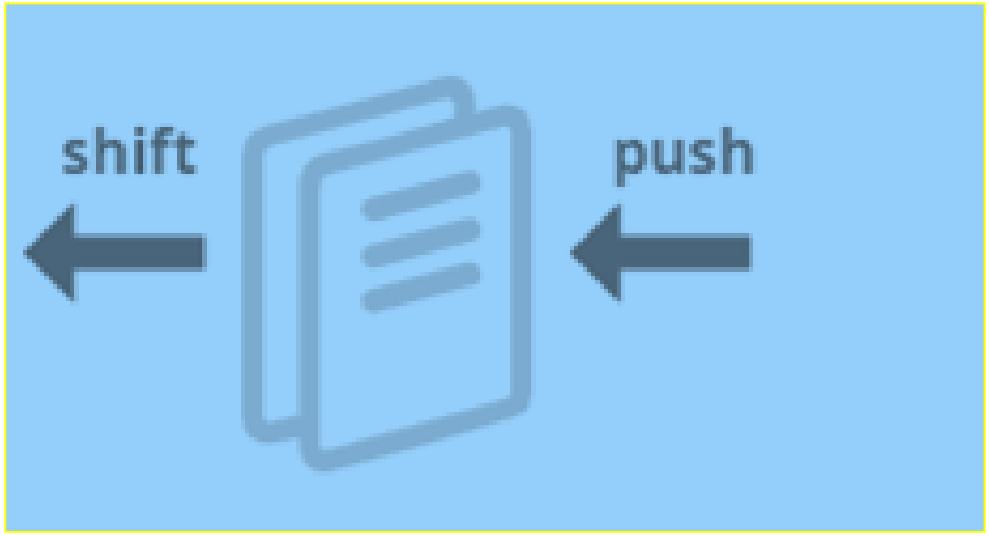
- Τα arrays στη JavaScript μπορούν να λειτουργήσουν τόσο ως **queues** όσο και ως **stacks**
- Επιτρέπουν έτσι την πρόσθεση/αφαίρεση στοιχείων **και από την αρχή και το τέλος**

- **stacks**, latest pushed item is received first LIFO (Last-In-First-Out)
- **queues**, FIFO (First-In-First-Out).

Queue

- **push** προσθέτει ένα στοιχείο στο τέλος
- **shift** εξάγει ένα στοιχείο από την αρχή
  - Έτσι το 2ο στοιχείο γίνεται 1ο
- **unshift** προσθέτει στοιχείο στην αρχή του πίνακα

Queue



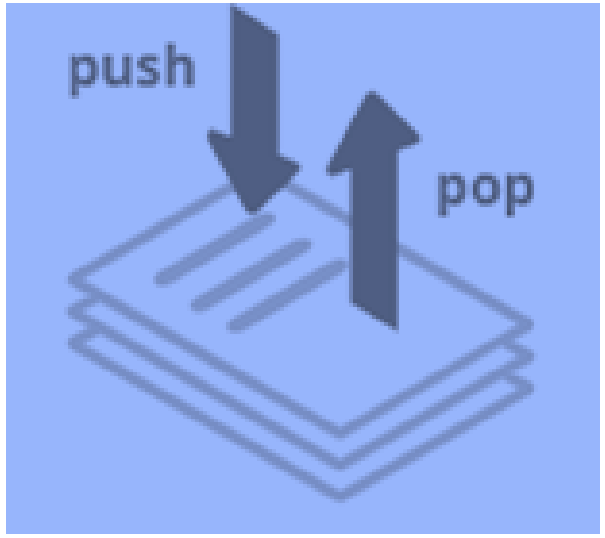
Queue

```
//push
var fruits = ["Apple", "Orange"];

fruits.push("Banana");

alert( fruits ); // Apple, Orange, Banana

//shift

alert( fruits.shift() ); // remove Apple and alert it
alert( fruits ); // Orange, Banana

//unshift

fruits.unshift('Apple');
fruits.unshift('pear');

alert( fruits ); // pear, Apple, Orange, Banana
</script>
```

stack

- push : προσθέτει ένα στοιχείο στο τέλος της στοίβας
- pop : εξάγει ένα στοιχείο από το τέλος στοίβας

stack



stack

```
//pop
var fruits = ["Apple", "Orange","Banana"];

alert( fruits.pop() ); // remove "Banana" and alert it

alert( fruits ); // Apple, Orange

//push

fruits.push("Banana");

alert( fruits ); // Apple, Orange, Banana
```

JS Arithmetic Operators

- Arithmetic Operators : + , - , * , / , % , ++ , --
- Comparison Operators: ==, !=, >, <, >=,<=
- Logical (or Relational) Operators : &&, ||, !
- Assignment Operators: =, +=, -=, *=, /= , %=
- **Conditional (or ternary) Operators: ? :**

  If Condition true? Then value X : Otherwise value Y

  Lets see an example

Lets understand operators better…

$$X=5$$

==	equal to	x == 8	false
		x == 5	true
		x == "5"	true
===	equal value and equal type	x === 5	true
		x === "5"	false
!=	not equal	x != 8	true
!==	not equal value or not equal type	x !== 5	false
		x !== "5"	true
		x !== 8	true
>	greater than	x > 8	false
<	less than	x < 8	true
>=	greater than or equal to	x >= 8	false
<=	less than or equal to	x <= 8	true

Source w3schools

Assignment Operators

Operator	This is the same as :	this
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
^=	x ^= y	x = x ^ y

Interesting sources

https://javascript.plainenglish.io/why-does-javascript-have-both-null-and-undefined-6a42fcca9301

Check deepFreeze function: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/freeze#what_is_shallow_freeze

https://medium.com/@ashfaqueahsan61/time-complexities-of-common-array-operations-in-javascript-c11a6a65a168


To be continued...