

# JavaScript variables

# 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 & Symbol**
  - **Μηδεν & 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 3 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

} όπως με το Pet

# Const

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

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

✘ Uncaught SyntaxError: Missing initializer in const declaration

# 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.



# 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
- Ωστόσο, αν δεν χρησιμοποιηθεί η λέξη -κλειδί var τότε η μεταβλητή θα πρέπει να αρχικοποιηθεί κατά τη δήλωσή της.
- 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)

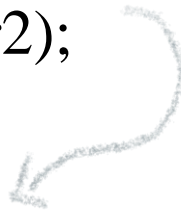
- Μέσα σε ένα string μπορούν να υπάρχουν επιπλέον εισαγωγικά αρκεί να διαφέρουν από αυτά που το ορίζουν:

```
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)

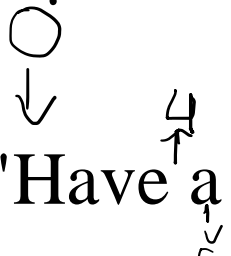
Τι περιμένουμε να δούμε στην κονσόλα στο παρακάτω παράδειγμα?

```
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.

To be continued...