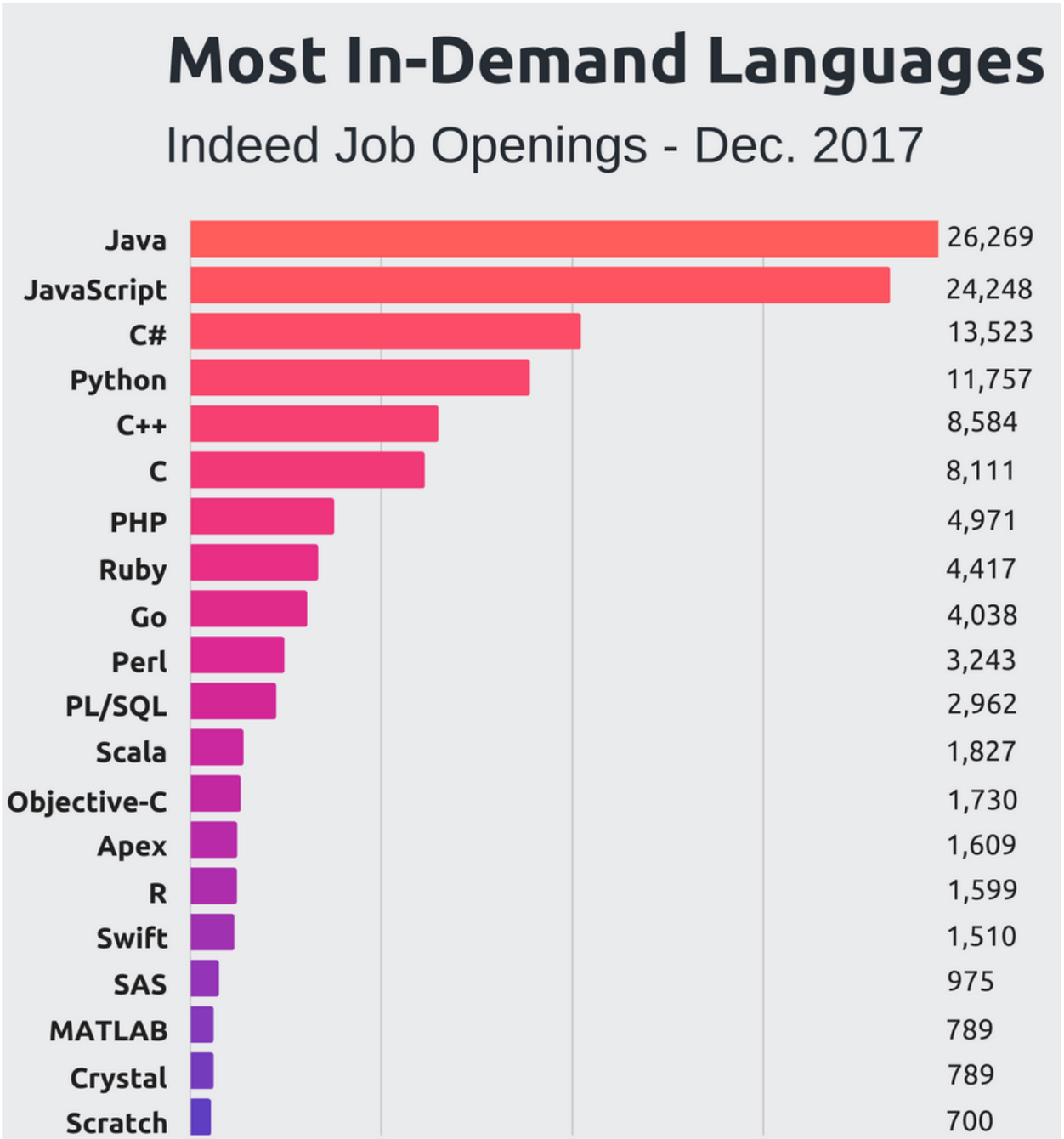
ΤΕΙ Κρήτης
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

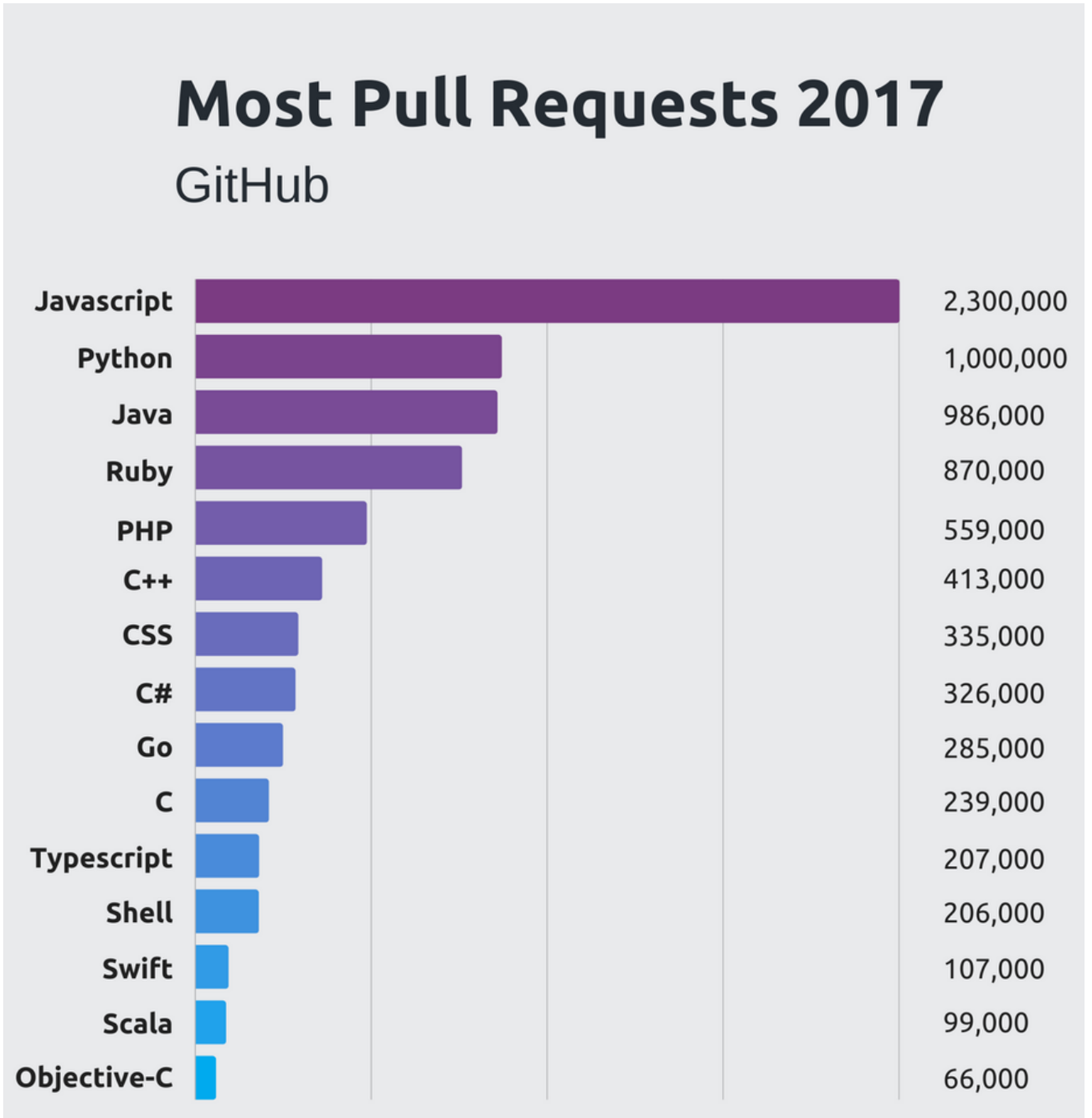
Θέματα Προγραμματισμού Διαδικτύου
~ Javascript ~

Στελιος Σφακιανάκης
Εαρινό 2020

Τί είναι;

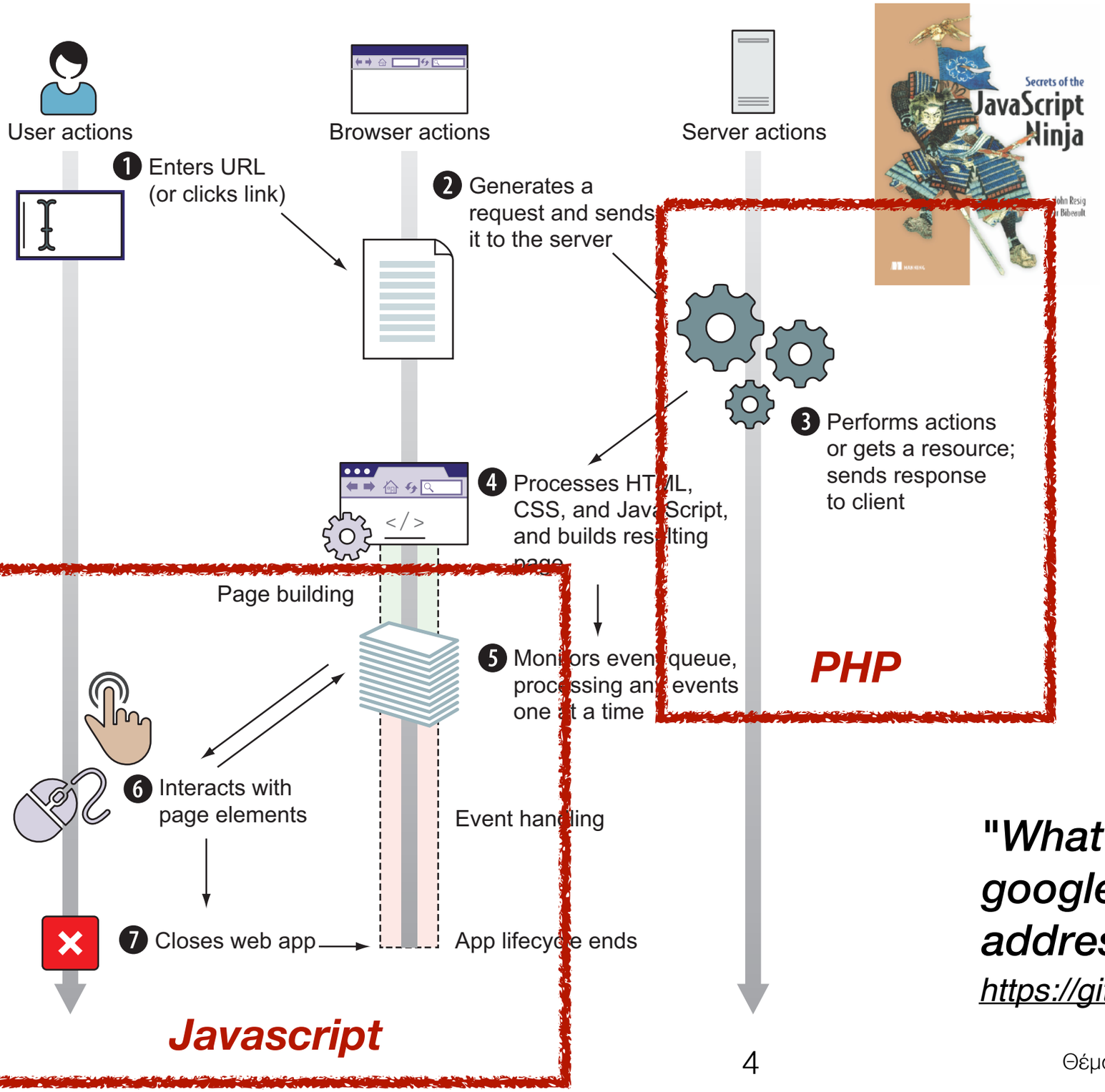
- Γλώσσα Προγραμματισμού για τη δημιουργία “**δυναμικών**” σελίδων στον Παγκόσμιο Ιστό.
  - “*Δυναμική*”, “*διερμηνευόμενη*” (interpreted), με “*ασθενείς τύπους*” (weakly typed)
- Η πιο δημοφιλής και ευρέως χρησιμοποιούμενη γλώσσα προγραμματισμού
  - Καθε πλοηγητής ιστού (web browser) διαθέτει μια μηχανή εκτέλεσης Javascript
  - “Ακόμα και η γιαγιά σας την χρησιμοποιεί!”
- Δημιουργήθηκε (μέσα σε 10 μέρες!) το 1995 από τον **Brendan Eich** του Netscape Communications
- *Δεν έχει καμία σχέση με την Java!!* Ονομάστηκε έτσι για εμπορικούς λόγους





https://stackify.com/popular-programming-languages-2018/

Η γενική εικόνα



Σχετικό:

"What happens when you type google.com into your browser's address box and press enter?"

https://github.com/alex/what-happens-when

Η Javascript ως γλώσσα προγραμματισμού*

* Με έμφαση στην μοντέρνα και στάνταρτ έκδοση της, "ECMAscript 2015" (ES-6)
Reference and Tutorials: https://developer.mozilla.org/en-US/docs/Web/JavaScript
Επίσης : http://speakingjs.com/es5/

Πριν προχωρήσουμε: Javascript vs ECMAScript

- Όλοι τη λέμε JavaScript, αλλά το όνομα της ειναι trademark της Oracle (που το "κληρονόμησε" από τη Sun)

- Οπότε, το official όνομα της ειναι **ECMAScript** που προτυποποιείται από τον οργανισμό ECMA

- Υπαρχουν διαφορετικές "εκδόσεις" της προδιαγραφής (specification) της ECMAScript

  - Από τις πρόσφατες εκδόσεις, περισσότερο ευρέως υποστηριζόμενη ειναι η 6η εκδοση ECMAScript 2015 ή ECMAScript 6

  - Υποστήριξη από πλοηγητές : http://kangax.github.io/compat-table/es2016plus/

Βασικό συντακτικό

```
// Σχόλιο μονής γραμμής
/*
 Σχόλιο που μπορεί να περιέχει
 περισσότερες από
 μια γραμμές
*/
var x;  // Ορισμός μεταβλητής
let x;  // Το ίδιο (Ορισμός μεταβλητής)

x = 3 + y;  // ανάθεση τιμής σε μια μεταβητη `x`

foo(x, y);  // κλήση συνάρτησης `foo` με παραμέτρους `x` και `y`
obj.bar(3);  // κλήση μεθόδου `bar` του αντικειμένου `obj`

// υπο-συνθήκη έλεγχος και εκτέλεση ενος μπλοκ εντολών
if (x === 0) {  // Εαν το `x` είναι ίσο με το μηδέν...
    x = 123;
}

// Δήλωση συνάρτησης `baz` με παραμέτρους `a` και `b`
function baz(a, b) {
    return a + b;
}
```

Μεταβλητές και Εντολές Εκχώρησης τιμής

- Όπως και σε άλλες γλώσσες προγ/μου οι μεταβλητές είναι "αποθήκες τιμών" (ονόματα θέσεων μνήμης που περιέχουν τιμές)

- Τα ονόματα των μεταβλητών στην Javascript αποτελούνται από Unicode χαρακτήρες: γράμματα, αριθμοί, και σύμβολα (π.χ. `$`, `_`) και είναι case-sensitive

  - Παραδείγματα: `var1`, `$myvar`, `_tst_`, ...

    - Κάποια ονόματα (π.χ. `true`, `false`, `class`, `try`, `catch`, `this`, ...) είναι δεσμευμένα και δεν μπορούν να χρησιμοποιηθούν ως ονόματα μεταβλητών

- Αλλάζουμε την τιμή μιας μεταβλητής με τον τελεστή εκχώρησης τιμής (`=`)

- Σε αντίθεση με την **C** αλλά όπως π.χ. και στην **PHP**, δεν δηλώνουμε τον τύπο δεδομένων μιας μεταβλητής (δηλ. αν είναι π.χ. αριθμός ή string)

Δήλωση Μεταβλητών

- Οι μεταβλητές εισαγονται/δηλώνονται με τα `var`, `let`, και `const` (χωρίς να δηλώνεται ο τύπος δεδομένων):
  - `const`, για σταθερές τιμές (π.χ. το π = 3,14..)
  - `let`, για "κανονικές" μεταβλητές

ES6

- Σε αντίθεση με το `var`, τα `let` και `const` είναι *scope* (block) aware οπότε και προτιμητέα
  - Το scope είναι εντολές που είναι διαχωρίζονται από τον υπόλοιπο κώδικα με αγκύλες

```
const name = "Stelios";
let age = 43;
age++; // 44
name = "Manos";// TypeError: Attempted to assign to readonly property.

if (true) {
   const name = "George";
   console.log(name); // George
}
console.log(name); // Stelios
age = "Foo" // No error, στο εξής το age θα είναι τύπου string με τιμή 'Foo'
```

Βασικοί Τύποι Δεδομένων

- Αριθμοί (numbers)
- Λογικοί (boolean)
- Αλφαριθμητικά (strings)
- Ειδικοί τύποι με αντίστοιχες τιμές: `null`, `undefined`

Πρωταρχικοί (primitive) τύποι

- Αντικείμενα (objects)

Αριθμοί

- **Number**: Κοινός τύπος για ακεραίους και αριθμούς κινητής υποδιαστολής! Δηλ. όλοι οι αριθμοί στην Javascript αναπαρίστανται εσωτερικά ώς doubles floating numbers.

- Ευρος τιμών μεταξύ $-(2^{53} - 1)$ και $2^{53} - 1$ (IEEE-754 64bits)

- Ειδικές τιμές : `+Infinity`, `-Infinity`, and `NaN` (not-a-number)
    - Π.χ. `1/0 == Infinity`,

- Δυαδική αναπαράσταση: `0b1011`, 16αδική: `0xBADCAFE3`

- Εκθετική αναπαράσταση : `2e10`, `6.626068E-34`

Αριθμητικοί τελεστές και μέθοδοι

- Οι συνήθεις τελεστές:
  - Αριθμητικοί: `+` , `-` , `*` , `/` , `%` , `**` (ύψωση σε δύναμη), και οι μοναδιαίοι `++` (αύξηση κατα ένα), `--` (μείωση κατά ένα)
  - Αριθμητικοί χειρισμού bits (not, and, or, xor, ..): `~`, `&`, `|`, `^`, `>>`, `<<`, `>>>`
- Το `Math` object παρέχει συνήθεις μαθηματικές συναρτήσεις, όπως `log`, `log10`, `exp`, `sqrt`, `abs`, τριγωνομετρικές (`cos`, `sin`, `tan`, ...), στρογγυλοποίησης (`round`, `floor`, `ceil`), κλπ
  - Πλήρης τεκμηρίωση στο http://mdn.io/math

Booleans

- **Boolean** με δύο τιμές: `true` και `false`.
- Γενικα, μη αληθείς τιμές θεωρούνται τα `false`, `0`, `""`, `null`, `undefined`, και `NaN`
- Τελεστές και πράξεις:
  - Οι συνήθεις λογικοί τελεστές: `&&` (and) `||` (or) `!` (not)
  - Τελεστές σύγκρισης: `<` , `>` , `>=` , `<=` , `==` , και οι τελεστές "αυστηρής" ισότητας: `===`, `!==`

Strings

- **Strings**: αλφαριθμητικά (πινακες από χαρακτήρες) με υποστήριξη Unicode, π.χ. `"Stelios"`, `"أنا بحبك"`, `'Maria 2019'`
  - Χρησιμοποιούνται μονά (`'`), διπλά (`"`), ή “ανάποδα μονά” (`` ` ``) εισαγωγικά
- Ως πίνακες απο χαρακτήρες, μπορούμε να προσπελάσουμε τον χαρακτήρα βασει της θέσης του (index), ξεκινώντας από το 0. Πχ. `"Stelios"[2] === 'e'`
- "Ενωση" (concatenation) δύο αλφαριθμητικών μέσω του τελεστή "+" ("πρόσθεση", όπως κάνουμε στην PHP με το `.`):
  - `"Hello " + "World"` μας δίνει το `"Hello World"`
- Χρήσιμες μέθοδοι: `indexOf()`, `charAt()`, `match()`, `search()`, `replace()`, `toUpperCase()`, `toLowerCase()`, `slice()`, `substr()`, ...

ES6 Template strings

- Επιτρέπουν την ενσωμάτωση μεταβλητών και εκφράσεων μέσα σε αλφαριθμητικά
  - (Αντίστοιχο του `"Hello $name"` της **PHP**)
  - Επιτρέπουν αλφαριθμητικά πολλαπλών γραμμών
  - Περικλείονται με το backtick (`)

```
let name = "George";
let age = 43;
let message = `Hi ${name}, next year you will be ${age+1} years old`;
```

Αντικείμενα

- Η Javascript είναι αντικειμενοστραφής!
- Όλα τα non-primitive values είναι objects (δηλ. ό,τι δεν είναι αριθμός, boolean, κλπ)
- Τα αντικείμενα είναι σύνθετοι τύποι, και μπορεί να είναι:
  - Απλά αντικείμενα, που τα εισάγουμε με αγκύλες (`{..}`)
  - Ημερομηνίες (`Date`)
  - Κανονικές εκφράσεις (`RegExp`)
  - `Set`, `Map` (νεα στην EcmaScript 6): Συνολα και "Λεξικά"
  - Ή ακόμα αντικείμενα που ορίζουμε εμείς!

Αντικείμενα

- Τα αντικείμενα είναι mappings μεταξύ “κλειδιών” (*keys*, *properties*) και τιμών (αντίστοιχα με τα **Arrays** με indexes strings στην **PHP**!)
  - Τα κλειδιά είναι strings ενώ οι τιμές μπορεί να είναι ο,τιδήποτε
  - Τα κλειδιά δεν είναι απαραίτητο να έχουν εισαγωγικά εκτός ..αν είναι απαραίτητο :-) (π.χ. αν περιέχει το κενο)
- Μπορούμε να διαβάσουμε (get) ή να αλλάξουμε (set) την τιμή ενός property σε ένα αντικείμενο
- Αφαίρεση ενός property με το `delete`, και απαρίθμηση τους με το `Object.keys()`

```
var person = {name: "Maria",
              "e-mail": "mar@d.com",
               age: 25};

person["e-mail"];  // "mar@d.com"

person.name; // "Maria"
person.age; // 25

delete person.age;

Object.keys(person); // [ 'name', 'e-mail' ]
```

Πίνακες (Κλασσικά Arrays)

- **Arrays** (πίνακες): αριθμημένες λίστες από "πράγματα", σε σειρά ξεκινώντας από το 0
  - `arr = [...]` ή `arr = new Array(...)`

- Για να προσπελάσουμε ένα στοιχείο χρησιμοποιούμε τη θέση του (index): `arr[1]` -> 2ο στοιχείο στο array

- Το `.length` δίνει το πλήθος των στοιχείων του array

Μέθοδοι των Arrays

- `arr.join(str)`: "ενώνει" όλα τα στοιχεία του array arr σε ενα string χωρισμένα με το string `str`

- `arr1.concat(arr2)`: επιστρέφει ενα νέο array που περιέχει όλα τα στοιχεία των `arr1` και `arr2`

- `arr.slice(j, k)`: επιστρέφει ενα νέο array που περιέχει όλα τα στοιχεία του `arr` από τη θέση `j` μέχρι και τη θέση πριν το `k` (δηλ. το `k-1`). Αν λείπει το `k` τότε παίρνει μεχρι το τέλος, ενώ αν δεν υπάρχει ούτε το `j` τότε επιστρέφει ένα αντίγραφο όλου του array.

- `arr.sort()`: ταξινομεί το array "in place" (δηλ. το τροποποιεί)

Άλλες μέθοδοι

- Μέθοδοι που δουλεύουν στο "τέλος" του array:
  - `arr.push(elem)` : εισαγωγή στοιχείου `elem` στο τέλος του `arr`
  - `arr.pop()` : εξαγωγή του τελευταίου στοιχείου
- Μέθοδοι που δουλεύουν στην αρχή του array:
  - `arr.shift()` : εξαγωγή του πρώτου στοιχείου
  - `arr.unshift(elem)` : εισαγωγή στοιχείου `elem` στην αρχή του `arr`

Πίνακες (Arrays)

```
fruits = ["Apple", "Orange", "Plum"];

alert( fruits.length );  // 3

alert( fruits[0] );      // Apple
alert( fruits[1] );      // Orange
alert( fruits[2] );      // Plum

fruits[2] = 'Pear';    // τώρα ["Apple", "Orange", "Pear"]
fruits.push('Melon'); // τώρα ["Apple", "Orange", "Pear", "Melon"]

fruits.shift();               // τώρα ["Orange", "Pear", "Melon"]
fruits.unshift('Avocando'); // τώρα ["Avocando","Orange", "Pear", "Melon"]
```

Δομές ελέγχου

Δομές Ελέγχου - Conditionals

- Υπό Συνθήκη (Conditionals): `if/else` και `switch/case`
  - Όπως και στη C, C++, Java, ...
- Στην if χρησιμοποιούμε μια λογική έκφραση για να επιλέξουμε ποιό από τα δύο θα κάνουμε (αν είναι `true` ή `false`)
  - Το τί θα κανουμε μπορεί να είναι περισσότερες από μία εντολές οπότε περικλείονται μέσα σε αγκύλες
- Στην `switch` ελέγχεται η τιμή μιας έκφρασης αν είναι ίση με κάποια από ένα σύνολο τιμών
- Υπάρχει ακόμα και ο τριαδικός τελεστής "`<condition> ? <then> : <else>`" για το υπολογισμό μιας υπο συνθήκη έκφρασης

```
if (myvar === 0) {
    // τότε
}

if (myvar === 0) {
    // τότε
} else {
    // αλλιώς
}

if (myvar === 0) {
    // τότε
} else if (myvar === 1) {
    // αλλιώς αν
} else if (myvar === 2) {
    // αλλιως-αν
} else {
    // αλλιώς
}
```

```
switch (fruit) {
    case 'banana':
        // ...
        break;
    case 'apple':
        // ...
        break;
    default:  // όλες οι άλλες περιπτώσεις
        // ...
}
```

Επαναληπτικές Δομές Ελέγχου

- Όμοίως με γνωστές γλώσσες προγ/μού η Javascript προσφέρει:
  - `for` επαναλήψεις: για έναν γνωστό αριθμός φορών, π.χ. από μια αρχική ακέραια τιμή σε μια τελική.
  - `while`: "όσο ισχύει μια συνθήκη, επανάλαβε"
  - `do..while`: "επανάλαβε όσο ισχύει μια συνθήκη" (δηλ. έλεγχος στο τέλος) [αυτό σημαίνει ότι η επάναληψη θα γίνει τουλάχιστον 1 φορα!]
- Μεσα σε μια επανάληψη μια εντολή `break` προκαλεί τον (πρώιμο) τερματισμός των επαναλήψεων ενώ μια εντολή `continue` προκαλεί την παράλειιψη της τρέχουσας επανάληψης και την μεταπήδηση στην επόμενη.

Παράδειγμα: απλή επανάληψη στα στοιχεία ενός Array

```
for (var i=0; i < arr.length; i++) {
    console.log(arr[i]);
}
```

Bug! Τι θα γίνει αν το array δεν έχει στοιχεία;;

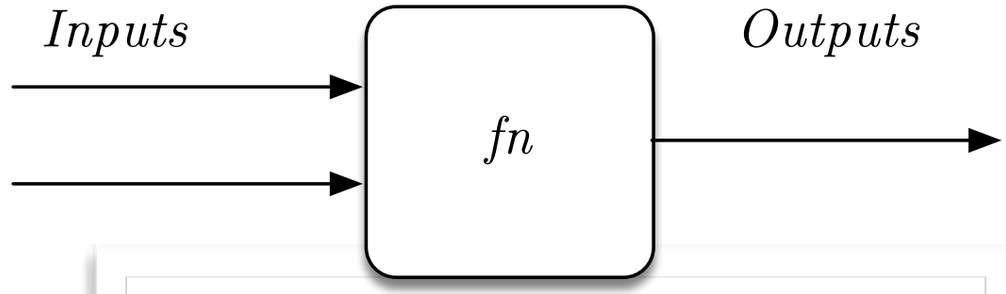
```
var i = 0;
while (i < arr.length) {
    console.log(arr[i]);
    i++;
}
```

```
var i = 0;
do {
console.log(arr[i]);
    i++;
} while (i < arr.length);
```

Συναρτήσεις

Δήλωση και κλήση συναρτήσεων: Γενικά

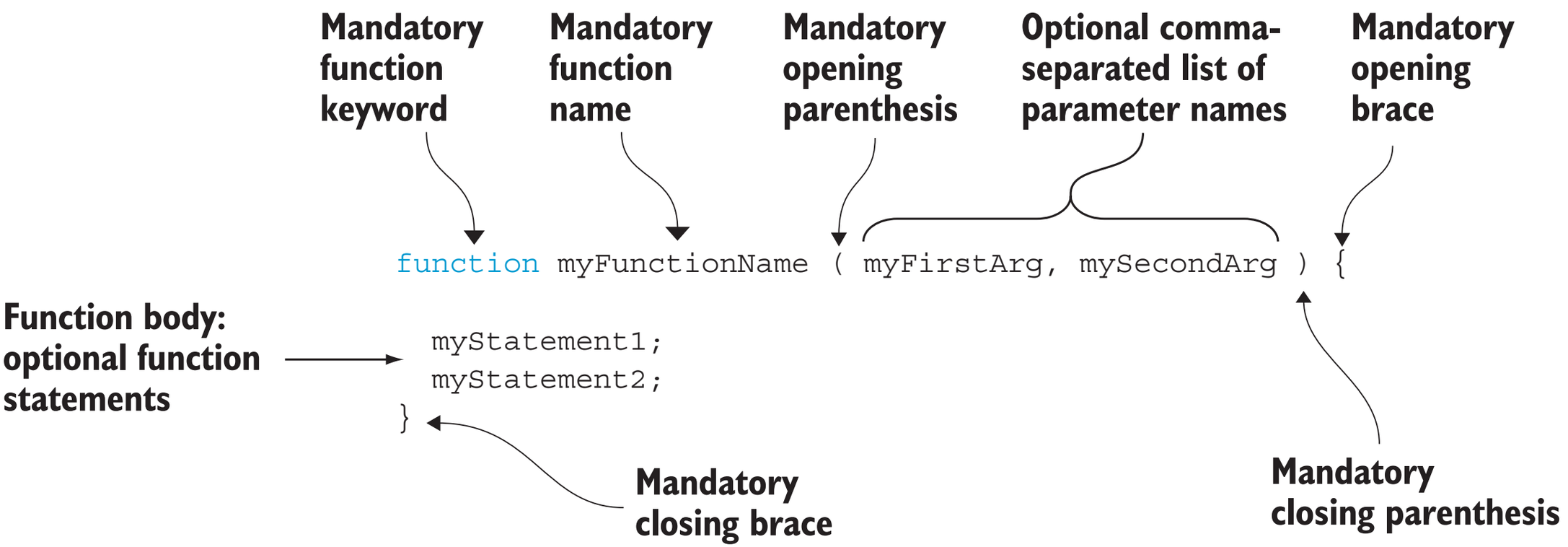
- Δήλωση της συνάρτησης με το keyword `function`
  - Παράμετροι μέσα σε παρενθέσεις
  - Εντολές συνάρτησεις μέσα σε αγκύλες
  - Το `return` προκαλεί την έξοδο από τη συνάρτηση και την επιστροφή της τιμής (αποτελέσματος) της συνάρτησης στο σημείο που κλήθηκε
- Κλήση της συνάρτησης με το όνομα της και τιμές εισόδου για τις παραμέτρους της




```
function is_even(n) {
    var rem = n % 2;
    return rem === 0;
}
function is_odd(n) {
    return !is_even(n);
}
```

```
var b = is_odd(933);
```

Ορισμός συναρτήσεων



Συναρτήσεις ως τιμές σε μεταβλητές

- Μπορούμε να δημιουργήσουμε "εκφράσεις συναρτήσεων" (*function expressions*) και να τις δώσουμε ως τιμές σε μεταβλητές

```
var is_even = function (n) {
    var rem = n % 2;
    return rem === 0;
}

var is_odd = function (n) {
    return !is_even(n);
}

var b = is_odd(933);
```

Callbacks

- Μια συνάρτηση που δίνεται ως όρισμα σε μια άλλη ονομάζεται **callback**
  - Είναι δηλαδή μια συνάρτηση που θα κληθεί αργότερα
- Η Javascript από κατασκευής υποστηρίζει "ασύγχρονο" προγραμματισμό με χρήση των callbacks
- Για παράδειγμα, η `setTimeout` έχει παράμετρο ενα callback function που θα κληθεί μετά από δοσμένο αριθμό milliseconds

```
setTimeout(function() {
    console.log("beep!");
}, 5000); // 5 seconds
```

Javascript και Web

Javascript και HTML

- Ο κώδικας Javascript πρέπει να ειναι μέσα στο `<script>` tag

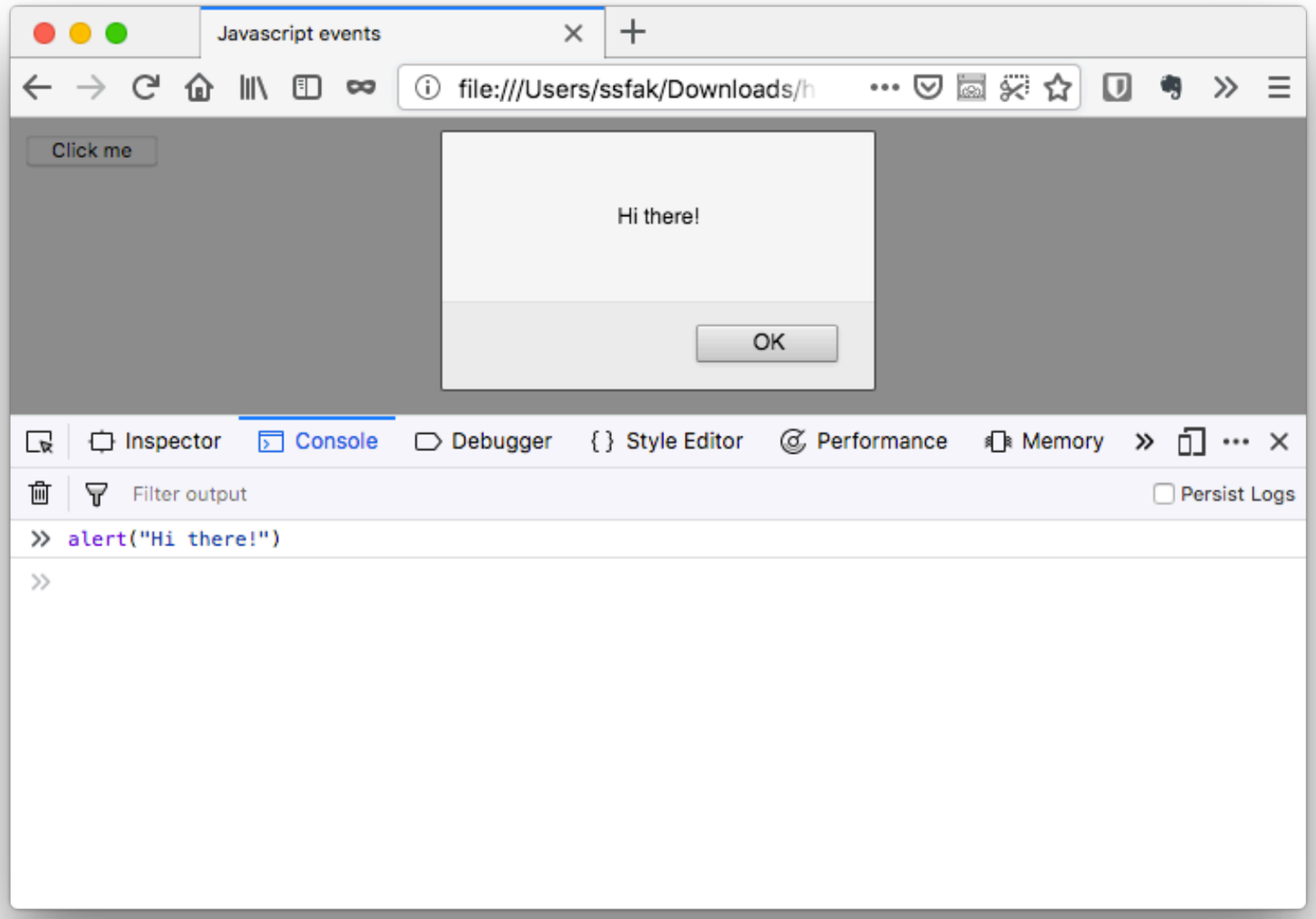
- Το `<script>` tag μπορεί να “αναφέρει” (κάνει reference με το `src` attribute) ένα ξεχωριστό αρχείο που περιέχει τον κώδικα:

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World in JavaScript</title>
</head>
<body>
    <script type="text/javascript">
        alert("Hello, world!");
    </script>
</body>
</html>
```

```
<script src="../js/lib.js"></script>
```

Browser developer tools

- Χρήσιμα για αποσφαλμάτωση:
  - `console.log` και `console.dir`
  - `alert` και `prompt`
- Εμφάνιση πληροφοριών για το DOM, stylesheets, επικοινωνία δικτύου, κλπ.
- Δυνατότητα τροποποίησης του περιεχομένου της σελίδας



Απλή επικοινωνία με τον χρήστη

- `alert(message)` : εμφανίζει το μήνυμα στο χρήστη

- `confirm(message)`: ζητά από το χρήστη να επιβεβαιώσει ή να ακυρώσει κάτι

- `prompt(msg, default)`: ζητά από το χρήστη να δώσει καποιο μήνυμα

Βασικά αντικείμενα με προγραμματιστική πρόσβαση

- `window` object: αναπαριστά την καρτέλα του πλοηγητή (browser tab) (π.χ. διαστάσεις)

- `navigator` object: η κατάσταση του browser (π.χ. προτιμώμενη γλώσσα χρήστη)

- `document` object: η σελίδα που βλεπουμε (HTML, CSS, ..) => **Document Object Model (DOM)**

Document Object Model

- Το DOM είναι μια ιεραρχική αναπαράσταση στη μνήμη του περιεχομένου μιας HTML (ή XML) σελίδας / εγγράφου

- Ορίζεται και μια "Προγραμματιστική Διεπαφή" (Programming Interface) για το χειρισμό των περιεχομένων

- Δηλ. με Javascript μπορούμε να χειριστούμε "δυναμικά" το περιεχόμενο, π.χ. να προσθέσουμε στοιχεία HTML, να αλλάξουμε τα περιεχόμενα τους κλπ

https://software.hixie.ch/utilities/js/live-dom-viewer/saved/6342

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Παράδειγμα σελίδας</title>
  </head>
  <body>
     <h1>Επικεφαλίδα</h1>
      <div>
        <img style="height:100px" src="https://media.giphy.com/...." alt="Bengal Tiger">
        <p>Μπορείτε να δείτε online το DOM στο <a href="https://...">Live DOM Viewer</a></p>
      </div>
  </
</ht
```

DOM Ιεραρχία

```
DOCTYPE: html
HTML
  HEAD
    #text:
    META charset="utf-8"
    #text:
    TITLE
      #text: Παράδειγμα σελίδας
    #text:
  #text:
  BODY
    #text:
    H1
      #text: Επικεφαλίδα
    #text:
    DIV
      #text:
      IMG style="height:100px" src="https://media.giphy.com/media/2Bkqvz4hcANy0/giphy-downsized.gif" alt="Bengal Tiger"
      #text:
      P
        #text: Μπορείτε να δείτε online το DOM στο
        A href="https://software.hixie.ch/utilities/js/live-dom-viewer/"
          #text: Live DOM Viewer
        #text:
      #text:
```

DOM Programming Interface

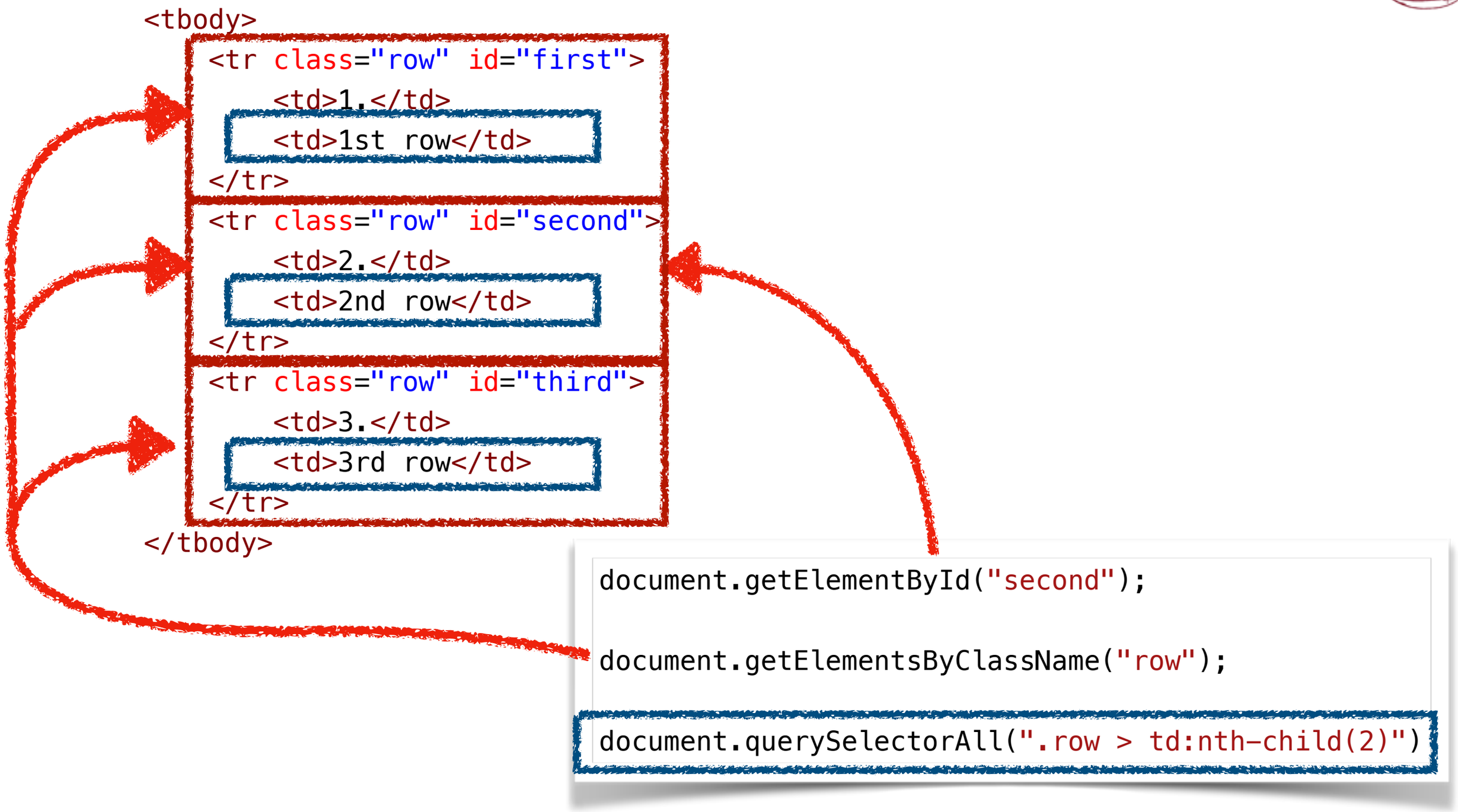
- Όλα τα HTML στοιχεία ορίζονται ως Javascript objects που έχουν ιδιότητες και μεθόδους
- Η ιεραρχία ξεκινάει από τη "ρίζα" που είναι το `window.document` object, και μετά ακολουθείται η δομή του HTML:
  - `window.document.head`
  - `window.document.body`

DOM object properties

- `nodeName` : το όνομα του στοιχείου (π.χ. "DIV") ή "#text" αν είναι κείμενο
- `textContent`: το περιεχόμενο κείμενο του στοιχείου (και όλων των στοιχείων που περιέχει)
- `innerHTML`: το HTML περιεχόμενο του στοιχείου
- Ιεραρχική δομή:
  - `parentNode` : γονικό στοιχείο,
  - `nextSibling`, `previousSibling` : επόμενος και προηγούμενος "αδελφός"
  - `firstChild`, `lastChild` : πρώτο και τελευταίο στοιχείο "παιδί"
  - `childNodes` : λίστα με παιδιά

Μεθοδοι "Εύρεσης" στοιχείων

Μέθοδος	Περιγραφή
`document.getElementById(id)`	Βρίσκει το στοιχείο με αυτό το id
`document.getElementsByTagName(name)`	Βρίσκει στοιχεία με αυτό το όνομα
`document.getElementsByClassName(name)`	Βρίσκει στοιχεία με αυτή το CSS class
`document.querySelector(selectors);`	Βρίσκει το πρώτο στοιχείο στο έγγραφο χρησιμοποιώντας τον CSS "επιλογέα" (selector) που δίνεται
`document.querySelectorAll(selectors);`	Βρίσκει μια λίστα με στοιχεία στο έγγραφο χρησιμοποιώντας τον CSS "επιλογέα" (selector) που δίνεται


<tbody>
<tr class="row" id="first">
<td>1.</td>
<td>1st row</td>
</tr>
<tr class="row" id="second">
<td>2.</td>
<td>2nd row</td>
</tr>
<tr class="row" id="third">
<td>3.</td>
<td>3rd row</td>
</tr>
</tbody>
document.getElementById("second");
document.getElementsByClassName("row");
document.querySelectorAll(".row > td:nth-child(2)")

Αλλάζοντας τα στοιχεία

Μέθοδος	Περιγραφή
`element.innerHTML = ...`	Αλλάζει το εσωτερικό HTML του στοιχείου
`element.setAttribute (attribute, value)`	Αλλάζει την τιμή ενός attribute του στοιχείου
`element.style.<property> = new style`	Αλλάζει το στυλ (style) ενός στοιχείου

- Π.χ. αν θέλουμε να "κρύψουμε" (να κάνουμε αόρατο) ένα element μπορούμε να γράψουμε:

- `elem.style.display="none"`

- `elem.style.display=""` για να ξαναγίνει "ορατό"

Προσθέτοντας και Αφαιρώντας Στοιχεία

Μέθοδος	Περιγραφή
`document.createElement(element)`	Δημιουργεί ένα HTML element
`parent.removeChild(element)`	Αφαιρεί ("σβήνει") ένα HTML element που είναι "παιδί" στο *parent*
`parent.appendChild(element)`	Προσθέτει ένα HTML element ως τελευταίο παιδί στο *parent*
`parent.replaceChild(newChild, oldChild)`	Αντικαθιστά ένα "παιδί" HTML element με ένα άλλο element στο *parent*
`document.write(text)`	Γράφει το κείμενο στο HTML output stream

Javascript και "γεγονότα" (Events)

- Ένα Event είναι ένα σήμα ότι κάτι συνέβη. Όλα τα στοιχεία του DOM δημιουργούν τέτοια σήματα αλλά events δημιουργούνται και από αλλού (π.χ. timers)

- Μπορούμε να αντιδράσουμε στα events εγκαθιστώντας (κάνοντας register) έναν handler δηλ. μια Javascript συνάρτηση που τρέχει όταν συμβεί το event.

Παραδείγματα Events

- Πληκτρολόγιο:
  - `keydown`: ένα πλήκτρο πατήθηκε
  - `keyup`: ένα πλήκτρο ελευθερώθηκε
- Ποντίκι:
  - `click` / `dblclick`: το ποντίκι έκανε "κλικ" / "διπλό κλικ" σε ένα στοιχείο
  - `contextmenu`: "δεξί κλικ" σε ένα στοιχείο
  - `mousedown` / `mouseup`: όταν ένα κουμπί του ποντικιού πατήθηκε/ελευθερώθηκε
  - `mouseover` / `mouseout`: όταν το ποντίκι "μπαίνει"/"βγαίνει" στην περιοχή ενός στοιχείου
- Document events:
  - `DOMContentLoaded`: όταν η σελίδα HTML έχει φορτωθεί και το DOM έχει δημιουργηθεί

https://developer.mozilla.org/en-US/docs/Web/Events

Event handlers

- Υπάρχουν διάφοροι τρόποι να ορίσουμε ποιος θα είναι ο handler σε ένα event:
  - Σε ένα HTML attribute (`on..`), π.χ.
    ```
    <button onclick="alert('Hello')">
    ```
  - Σε Javascript θέτοντας την κατάλληλη attribute του στοιχείου
    ```
    let b = document.getElementById('btn');
    b.onclick = function() { alert ('Hello'); });
    ```
  - (**Προτιμότερο**) Σε κώδικα Javascript με το `addEventListener`

Σύνδεση HTML στοιχείου με event handler με το addEventListener

```
<body>
    <button id="btn">Click me</button>
</body>

<script type="text/javascript">
        var button = document.getElementById('btn');
        button.addEventListener('click',
                                function () {
                                    alert("You clicked me!");
                                });
</script>
```

- Το `addEventListener` έχει πρώτο όρισμα το event type (‘`click`’, ‘`load`’, … δηλ. χωρίς το `on`) και 2ο τη συνάρτηση (event handler/callback)

Παράδειγμα σύνδεσης μέσω attribute

```html
<!DOCTYPE html>
<html>

<head>
    <title>Javascript events</title>
</head>

<body>
    <h1>Welcome!</h1>
    <button onclick="message()">Press me!</button>
</body>

<script type="text/javascript">
    let count = 0;

    function message() {
        count++;
        alert(`You have pressed me ${count} times!`);
        document.querySelector("button").innerHTML =
            `Press me once more for ${count+1} presses`;
    }
</script>

</html>
```