ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΟ ΠΡΟΤΥΠΟ WEBASSEMBLY

Καρέλας Βασίλειος

Επιβλέπων καθηγητής: **Κασκάλης Θεόδωρος**

Φεβρουάριος 2021

Επισκόπηση της Παρουσίασης

- Πρόβλημα
- Σκοπός και στόχοι
- Θεωρητικό υπόβαθρο
- *Προηγούμενες προσπάθειες*
- *Το πρότυπο WebAssembly*
- *Συγκριτική μελέτη*
- *Συμπεράσματα*

Πρόβλημα (1/1)

- Διαδικτυακές εφαρμογές με χαμηλές επιδόσεις ως προς την ταχύτητα.
- Ευρεία χρήση τεχνολογιών οι οποίες αναπτύχθηκαν αρχικά για διαφορετικές χρήσεις και υπό άλλες συνθήκες.
- Οι υπάρχουσες τεχνολογίες αδυνατούν ίσως να ακολουθήσουν πλήρως την έκρηξη χρήσης του Διαδικτύου.

Σκοπός και στόχοι (1/2)

Σκοπός της διπλωματικής εργασίας:

Η διερεύνηση του προτύπου WebAssembly, των πλεονεκτημάτων τα οποία προσφέρει σε σχέση με άλλες τεχνολογίες και των προοπτικών που δημιουργούνται με την χρήση της.

Σκοπός και στόχοι (2/2)

Στόχοι:

- Παρουσίαση του προτύπου WebAsselmbly και του τρόπου με τον οποίο λειτουργεί, σε συνδυασμό με την χρήσης της γλώσσας JavaScript.
- Αναλυτική περιγραφή της διαδικασίας ανάπτυξης προγραμμάτων που κάνουν χρήση του προτύπου WebAssembly.
- Σύγκριση του προτύπου WebAssembly με άλλες τεχνολογίες που αναπτύχθηκαν για την βελτίωση της επίδοσης της Javascript.
- Μελέτη των επιδόσεων κάποιων προγραμμάτων που δημιουργήθηκαν με το πρότυπο WebAssembly για το σκοπό αυτό, σε σχέση με αντίστοιχα προγράμματα που έχουν δημιουργηθεί με απλή Javascript.
- Διερεύνηση των χρήσεων που μπορεί να έχει η νέα αυτή τεχνολογία και η επίδρασή της στη λειτουργία του Παγκόσμιου Ιστού.

Θεωρητικό υπόβαθρο (1/6)

Javascript

- Γλώσσα προγραμματισμού για ηλεκτρονικούς υπολογιστές η οποία κάνει χρήση *διερμηνευτή* (interpreter) για την ανάπτυξη σεναρίων (scripts).
- Αναπτύχθηκε τη δεκαετία του 1990 από τον Brendan Eich της Netscape, αρχικά με την επωνυμία *Mocha, για την ανάπτυξη κώδικα που μπορεί να ενσωματωθεί σε ιστοσελίδες.*
- Το Νοέμβριο του 1996, η Netscape έκανε υποβολή της γλώσσας για την κατοχύρωσή της ως βιομηχανικό πρότυπο στο Ecma International και κατοχυρώθηκε με το όνομα ECMAScript.
- Το 2021 είναι μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού.
- Το Διαδίκτυο δεν έχει καμία σχέση με την εποχή που αναπτύχθηκε η γλώσσα και οι απαιτήσεις του είναι τελείως διαφορετικές.

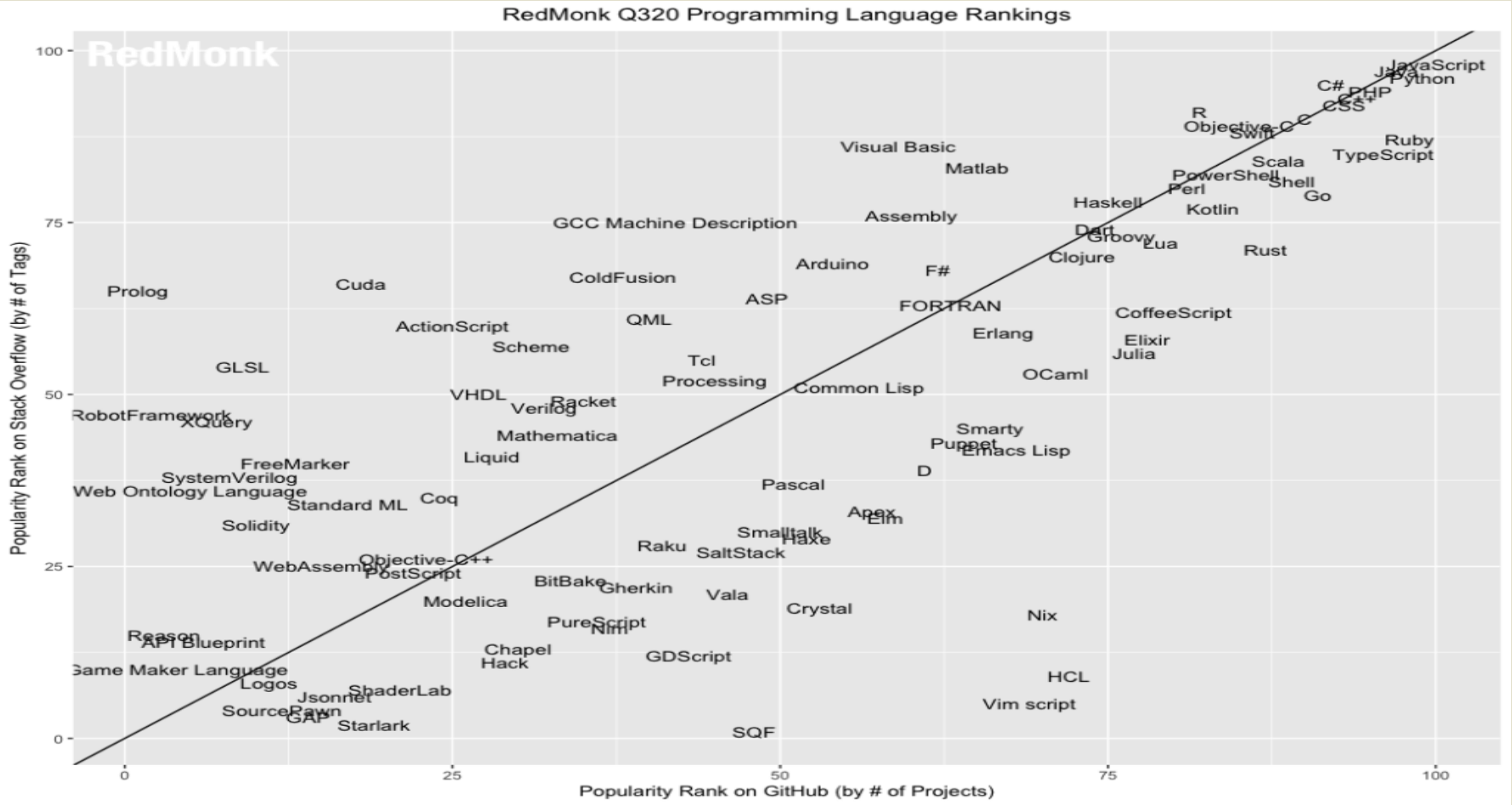
Θεωρητικό υπόβαθρο (2/6)

Χαρακτηριστικά Javascript

- Δομημένη γλώσσα υψηλού επιπέδου.
- Διερμηνευόμενη, κάνει χρήση interpreter. Αυτό την κάνει πιο αργή ως προς την εκτέλεση.
- Δυναμική γλώσσα. Μεταβλητές και συναρτήσεις μπορούν να αλλάξουν ή να δημιουργηθούν νέες κατά την εκτέλεση του προγράμματος.
- Ασθενείς τύποι μεταβλητών.
- Υποστηρίζει αντικειμενοστρεφή προγραμματισμό.
- Υποστηρίζει συναρτησιακό προγραμματισμό.
- Ανεξάρτητη από την πλατφόρμα (platform independent)

Θεωρητικό υπόβαθρο (3/6)

Δημοφιλέστερη γλώσσα με βάση τον δείκτη RedMonk που βασίζεται σε δεδομένα που λαμβάνονται από το GitHub και το Stack Overflow



Θεωρητικό υπόβαθρο (4/6)

Τρίτη θέση με βάση τον δείκτη PYPL που βασίζεται στη συχνότητα των αναζητήσεων που γίνονται στο Διαδίκτυο

Worldwide, Oct 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	31.02 %	+2.2 %
2		Java	16.38 %	-2.8 %
3		JavaScript	8.41 %	+0.4 %
4		C#	6.52 %	-0.6 %
5		PHP	5.83 %	-0.4 %
6		C/C++	5.56 %	-0.4 %

Θεωρητικό υπόβαθρο (5/6)

Χρήσεις της Javascript

- Προγραμματισμός στη μεριά του πελάτη (client-side scripting) για τη δημιουργία λειτουργικών και φιλικών ιστοσελίδων.
- Χρήση frameworks τα οποία επιτρέπουν την περισσότερο τυποποιημένη και εύκολη ανάπτυξη διεπαφών με τον χρήστη (GUI), **όπως** *React*, *Angular*, *Vue*, *Ember*, *Backbone* κλπ.
- Χρήση τεχνικών για την άμεση λήψη δεδομένων από έναν server (Ajax).
- Προγραμματισμός στη μεριά του εξυπηρετητή (server-side scripting) με τη βοήθεια του node.js.
- Δημιουργία αυτόνομων εφαρμογών με το Electron framework (π.χ. Atom).
- Ανάπτυξη εφαρμογών για “έξυπνες” συσκευές με το Apache Cordova.

Θεωρητικό υπόβαθρο (6/6)

Αδυναμίες της Javascript

- Έχει αρχίσει να χρησιμοποιείται για σκοπούς για τους οποίους δεν έχει σχεδιαστεί από την αρχή.
- Χρήση interpreter αντί για compiler.
- Χαμηλές επιδόσεις ως προς την ταχύτητα εκτέλεσης των εφαρμογών.
- Αδυναμία εκμετάλλευσης έτοιμων προγραμμάτων που έχουν δημιουργηθεί σε άλλες γλώσσες προγραμματισμού.
- Δυσκολίες να υποστηρίξει την τεράστια αύξηση χρήσης του Διαδικτύου.

Προηγούμενες προσπάθειες (1/4)

Στιγμιαία μεταγλώττιση (Just-In-Time compilation)

- Διαθέτουν αναλυτή (monitor ή profiler), που εξετάζει τον κώδικα, έτσι ώστε να εντοπιστούν οι εντολές οι οποίες επαναλαμβάνουν συχνά την εκτέλεσή τους.
- Διαφέρουν από υλοποίηση σε υλοποίηση.
- Χαρακτηρίζονται κάποια τμήματα κώδικα ως “θερμά” ή “καυτά”, ανάλογα με το κατά πόσο επαναλαμβάνονται.
- Οι “θερμές” εντολές μεταφράζονται εξ’ αρχής, ώστε να μη χάνεται χρόνος κατά την εκτέλεση.
- Γίνεται επιπλέον βελτιστοποίηση για τις “καυτές” εντολές.
- Η Javascript θέτει περιορισμούς, εξ’ αιτίας των οποίων θα πρέπει να γίνει κάποιες φορές deoptimization.
- Έχουμε βελτίωση στην απόδοση, αλλά όχι στον επιθυμητό βαθμό.

Προηγούμενες προσπάθειες (2/4)

asm.js

- Υποσύνολο της γλώσσας Javascript, το οποίο περιλαμβάνει μόνο κάποια επιθυμητά στοιχεία της γλώσσας, τα οποία επιδέχονται βελτιστοποιήσεων κατά την μετάφρασή τους.
- Ξεκίνησε από τη Mozilla και δίνει τη δυνατότητα αυτόματης μεταγλώττισης έτοιμου κώδικα C/C++ σε asm.js (μέσω εργαλείων όπως το Emscripten).
- Χρησιμοποιήθηκε τόσο για παιχνίδια, όσο και για εφαρμογές.
- Έλλειψη από πρότυπα τα οποία να υποστηρίζονται από όλους τους κατασκευαστές.
- Θέματα ασυμβατότητας.
- Μεγάλη αθυστέρηση κατά τη φόρτωση της εφαρμογής στο περιβάλλον του προγράμματος πλοήγησης.

Προηγούμενες προσπάθειες (3/4)

Native Client της Google

- Κάνει δυνατή την εκτέλεση του αντικείμενου κώδικα μίας εφαρμογής (μορφή x86, ARM ή MIPS,) που έχει δημιουργηθεί με άλλη γλώσσα προγραμματισμού, μέσα από το περιβάλλον ενός προγράμματος πλοήγησης.
- Εκτέλεση του κώδικα σε ελεγχόμενο περιβάλλον για λόγους ασφαλείας.
- Υποστήριξη παιχνιδιών και εφαρμογών.
- Έπρεπε να δημιουργούνται διαφορετικά εκτελέσιμα, ανάλογα με την αρχιτεκτονική.
- Με την τεχνολογία *Portable Native Client* λύθηκε το πρόβλημα της φορητότητας.
- Απόσυρση της τεχνολογίας από την Google για χάρη της WebAssembly.

Προηγούμενες προσπάθειες (4/4)

Emscripten και LLVM

- Η ιδέα της αυτόματης μεταγλώττισης από μία γλώσσα προγραμματισμού σε μία άλλη, είναι παλιά.
  - *Web Toolkit* της Google *για μετάφραση από Java σε Javascript*
  - *Pyjamas για μετάφραση από Python σε Javascript κτλ.*
- Με τον Emscripten είναι δυνατή η αυτόματη μεταγλώττιση προγράμματος σε Javascript από διάφορες γλώσσες προγραμματισμού, όπως οι C και C++.
- Στην ουσία μεταφράζει από την ενδιάμεση μορφή *LLVM* (Low-Level Virtual Machine), η οποία μπορεί να παραχθεί από τον compiler της αντίστοιχης γλώσσας.
- Χρησιμοποιείται και για την αυτόματη μεταγλώττιση ενός προγράμματος σε γλώσσα WebAssembly.

Το πρότυπο WebAssembly (1/11)

- Ανοικτό πρότυπο που περιγράφει τη δυαδική κωδικοποίηση εκτελέσιμων προγραμμάτων, με τη βοήθεια του οποίου μπορεί να διασφαλιστεί η φορητότητα, ασφάλεια και η ταχύτητα εκτέλεσης μίας εφαρμογής.
- Στόχος η ενσωμάτωση εφαρμογών υψηλών επιδόσεων μέσα σε ιστοσελίδες, με τη δημιουργία κατάλληλου περιβάλλοντος εκτέλεσης.
- Μπορεί να χρησιμοποιηθεί και για την ανάπτυξη αυτόνομων εφαρμογών.
- Ανακοινώθηκε το 2015 με την επίδειξη εκτέλεσης παιχνιδιού Angry Bots που δημιουργήθηκε με το Unity.
- Το WebAssembly Working Group έχει σαν μέλη τις μεγαλύτερες εταιρείες που σχετίζονται με την ανάπτυξη του Διαδικτύου.
- Υποστήριξη από νέες εκδόσεις Mozilla Firefox, Chrome, Microsoft Edge, Safari κτλ.

Το πρότυπο WebAssembly (2/11)

Υποστήριξη γλωσσών προγραμματισμού

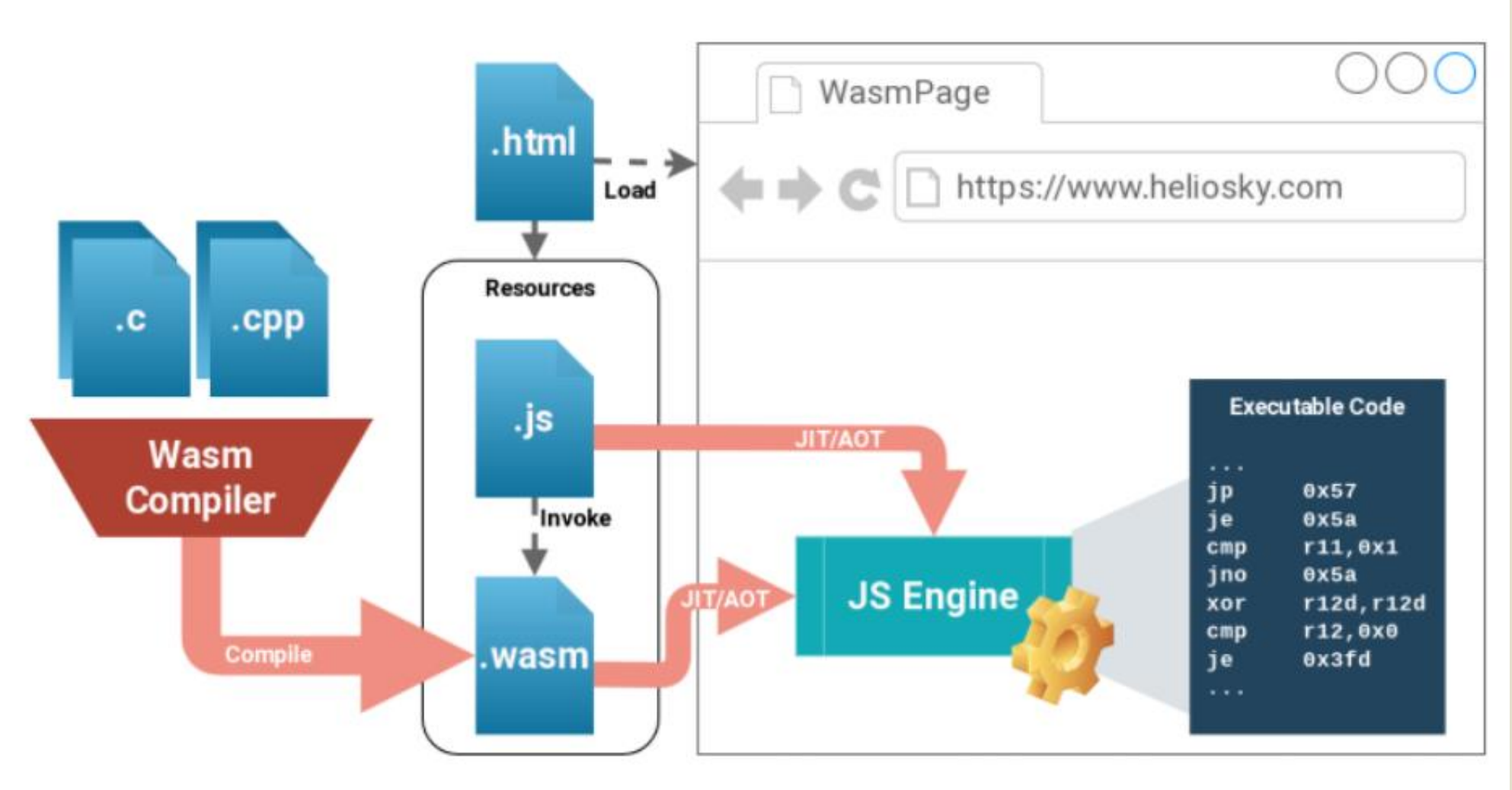
- C και C++ μέσω Emscripten.
- Rust, επίσημη υποστήριξη της γλώσσας.
- Go, επίσημη υποστήριξη της γλώσσας.
- C# πειραματική υποστήριξη
- Java, με τη βοήθεια του TeaVM
- Python, μέσω του Pyodide
- PHP, σε πειραματικό στάδιο

Το πρότυπο WebAssembly (3/11)

Εκτέλεση κώδικα WebAssembly

- Δημιουργία εκτελέσιμου κώδικα σε μορφή Wasm με τη βοήθεια κατάλληλων εργαλείων.
- Φόρτωση και εκτέλεση του παραγόμενου Wasm module μέσα στο περιβάλλον της ιστοσελίδας.
  - *Χρήση τεχνικών JIT (Just In Time) ή AOT (Ahead OF Time) για την μετατροπή σε δυαδική μορφή.*
- Έλεγχος κατά τη φόρτωση για ανάγκη επιπλέον βιβλιοθηκών.

Το πρότυπο WebAssembly (4/11)



Το πρότυπο WebAssembly (5/11)

Αρχιτεκτονική του προτύπου WebAssembly

- Χρήση αρχιτεκτονική σωρούς
- Κάθε εντολή χρησιμοποιεί ορίσματα που βρίσκονται αποθηκευμένα στην κορυφή μίας μνήμης σωρού.
- Αποθήκευση των αποτελεσμάτων πίσω στο σωρό.
- Τύποι δεδομένων:
  - *Ακέραιοι 32 και 64 bits*
  - *Πραγματικοί 32 και 64 bits*

Το πρότυπο WebAssembly (6/11)

Τύποι εντολών

Αριθμητικές εντολές	Εκτελούν αριθμητικές και λογικές πράξεις στους αριθμούς που περιέχονται στη λίστα ορισμάτων τους
Εντολές χειρισμού σωρού	Χειρίζονται τα περιεχόμενα του σωρού που χρησιμοποιείται για την αποθήκευση των ορισμάτων
Εντολές χειρισμού μεταβλητών	Διαβάζουν ή τροποποιούν τα περιεχόμενα των τοπικών και καθολικών μεταβλητών που χρησιμοποιεί το πρόγραμμα
Εντολές χειρισμού μνήμης	Διαβάζουν δεδομένα από τη μνήμη ή αποθηκεύουν αποτελέσματα σε αυτή. Γίνεται χρήση γραμμικής μνήμης.
Εντολές ελέγχου	Εντολές που ελέγχουν τη ροή του προγράμματος.

Το πρότυπο WebAssembly (7/11)

Καθολικές μεταβλητές

```
(module
  (import "env" "g1" (global $g1 i32))        ;; immutable
  (import "env" "g2" (global $g2 (mut f32))) ;; mutable

  (global $g3 (mut i32) (i32.const 123)) ;; mutable
  (global $g4 (mut i64) (i64.const 456)) ;; mutable
  (global $g5 f32 (f32.const 1.5))       ;; immutable
  (global $g6 f64 (f64.const 2.5))       ;; immutable

  (func $main
    ;; $g3 = $g1
    (global.get $g1)
    (global.set $g3)
  )
)
```

Το πρότυπο WebAssembly (8/11)

Τοπικές μεταβλητές

```
(module
  (func $main (param $a i32) (param $b f32)
    (local $c i32)
    (local $d i64)
    (local $e f32)
    (local $f f64)

    ;; $c = $a
    (local.get $a)
    (local.set $c)
  )
)
```

Το πρότυπο WebAssembly (9/11)

Συναρτήσεις

```
(module
  (func $main (export "main") (result i32)
    (call $max (i32.const 20) (i32.const 80))
  )
  (func $max (param $a i32) (param $b i32) (result i32)
    (local.get $a)
    (local.get $b)
    (i32.gt_s (local.get $a) (local.get $b))
    (select)
  )
)
```

Το πρότυπο WebAssembly (10/11)

Έλεγχος Ροής

```
(module
  (func $max (param $a i32) (param $b i32) (result i32)
    (i32.gt_s (local.get $a) (local.get $b))
    (if (result i32)
      (then (local.get $a))
      (else (local.get $b))
    )
  )
)
```

Το πρότυπο WebAssembly (11/11)

Δομή επανάληψης

```
(module
  (func $sum (param $from i32) (param $to i32) (result i32)
    (local $n i32)

    (loop $l
      ;; $n += $from
      (local.set $n (i32.add (local.get $n) (local.get $from)))
      ;; $from++
      (local.set $from (i32.add (local.get $from) (i32.const 1)))
      ;; if $from <= $to { continue }
      (br_if $l (i32.le_s (local.get $from) (local.get $to)))
    )

    ;; return $n
    (local.get $n)
  )
)
```

Συγκριτική μελέτη (1/7)

- Σύγκριση επιδόσεων εφαρμογών που έχουν αναπτυχθεί με τη βοήθεια της WebAssembly και της Javascript
- Η σύγκριση γίνεται ανάμεσα σε εφαρμογές WebAssembly και Javascript με παρόμοια λειτουργικότητα, που εκτελούνται στο ίδιο περιβάλλον.
- Η σύγκριση αφορά το συνολικό χρόνο από το άνοιγμα της ιστοσελίδας που περιέχει την εφαρμογή, μέχρι την τελική παρουσίαση των αποτελεσμάτων.
- Ο κώδικας WebAssembly δημιουργήθηκε με αυτόματη μετατροπή από κώδικα σε μορφή C, με το εργαλείο Emscripten.
- Η μέτρηση του χρόνου γίνεται με τη βοήθεια ειδικού πρόσθετου του Google Chrome.

Συγκριτική μελέτη (2/7)

Κώδικας C ακολουθίας Fibonacci

```c
int main() {
  return fibonacci(40);
}

int fibonacci(int n) {
  if (n == 0)
    return 1;
  if (n == 1)
    return 1;
  return fibonacci(n-1) + fibonacci(n-2);
}
```

Συγκριτική μελέτη (3/7)

Κώδικας WebAssembly (μορφή WAT)

```
(module
  (type $t0 (func))
  (type $t1 (func (result i32)))
  (type $t2 (func (param i32) (result i32)))
  (func $__wasm_call_ctors (type $t0))
  (func $main (export "main") (type $t1) (result i32)
    i32.const 40
    call $fibonacci)
  (func $fibonacci (type $t2) (param $p0 i32) (result i32)
    (local $l0 i32)
    i32.const 1
    set_local $l0
    block $B0
      get_local $p0
      i32.const 2
      i32.lt_u
      br_if $B0
      i32.const 1
      set_local $l0
      loop $L1
        get_local $p0
        i32.const -1
```

Συγκριτική μελέτη (4/7)

Κώδικας Javascript για τη φόρτωση του WebAssembly module

```
fetch('../out/main.wasm').then(response =>
  response.arrayBuffer()
).then(bytes => WebAssembly.instantiate(bytes)).then(results => {
  instance = results.instance;
  document.getElementById("container").textContent = instance.exports.main();
}).catch(console.error);
```

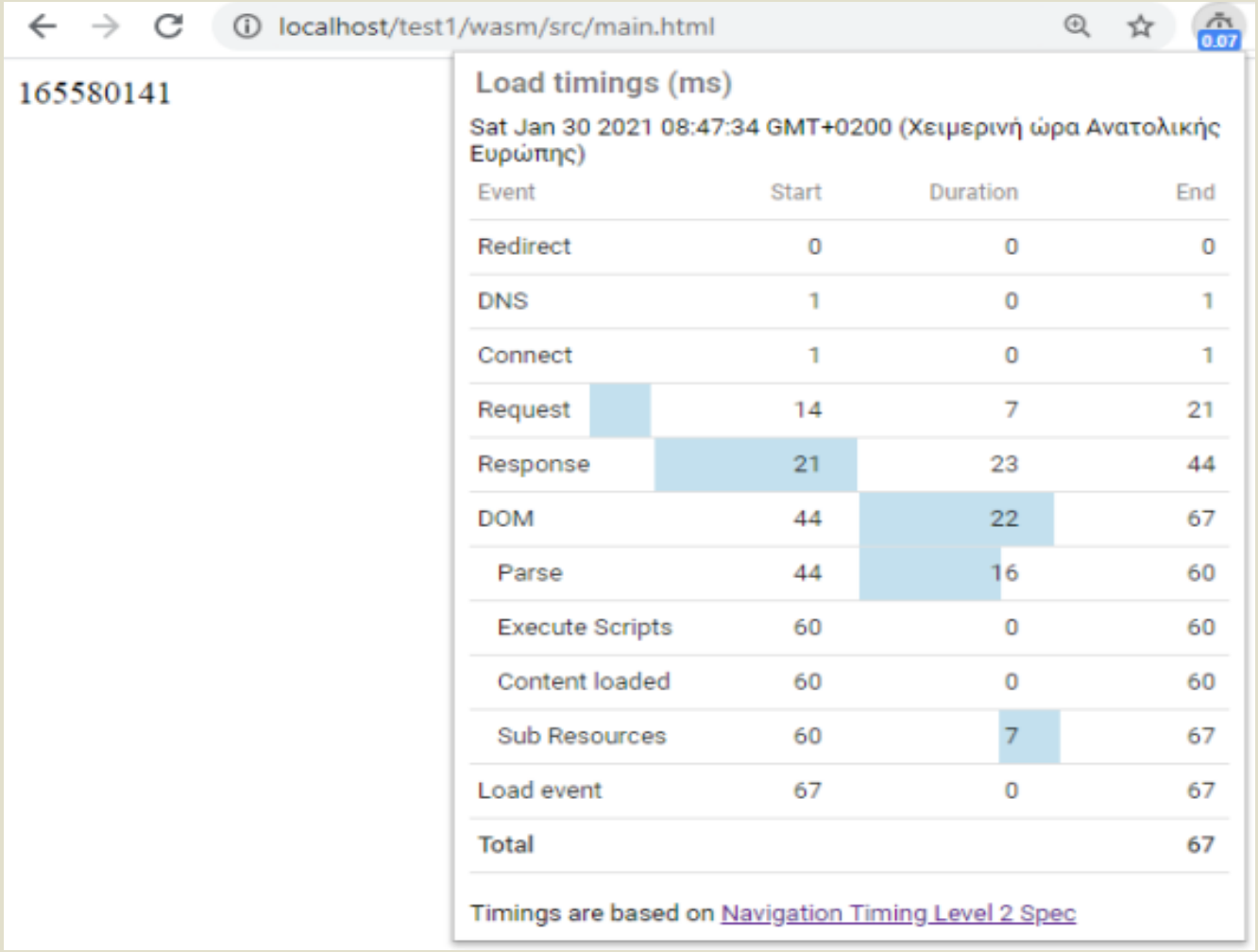
Συγκριτική μελέτη (5/7)

Κώδικας Javascript για την ακολουθία Fibonacci

```
function fibonacci(n) {
  if (n == 0)
    return 1;
  if (n == 1)
    return 1;
  return fibonacci(n-1) + fibonacci(n-2);
}
```

Συγκριτική μελέτη (6/7)

Μέτρηση χρόνου φόρτωσης με το πρόσθετο Page Load Time



Συγκριτική μελέτη (7/7)

Σύγκριση χρόνων για την εμφάνιση των αποτελεσμάτων

	Πρόγραμμα	Javascript	Wasm	Speedup
1	Ακολουθία Fibonacci	1,70	0,67	2,54
2	Παραγοντικό	1,46	0,56	2,61
3	Πρόσθεση ακεραίων	0,25	0,23	1,09
4	Πρόσθεση πραγματικών	0,33	0,29	1,14
5	Πολλαπλασιασμός ακεραίων	1,25	0,34	3,68
6	Πολλαπλασιασμός δεκαδικών	1,78	0,64	2,78
7	Ταξινόμηση ακεραίων	1,29	1,02	1,26
8	Ταξινόμηση δεκαδικών	1,78	1,35	1,31
9	Πολλαπλασιασμός πινάκων ακεραίων	2,15	1,43	1,50
10	Πολλαπλασιασμός πινάκων πραγματικών	2,59	1,78	1,46

Συμπεράσματα

- Στον Παγκόσμιο Ιστό χρησιμοποιούνται ευρέως κάποιες τεχνολογίες για ρόλους για τους οποίους δεν έχουν σχεδιαστεί, με αποτέλεσμα να παρουσιάζουν κάποιες αδυναμίες.
- Το πρότυπο WebAssembly είναι πολλά υποσχόμενο και συγκεντρώνει την υποστήριξη των μεγάλων εταιρειών του χώρου.
- Η WebAssembly μπορεί να βοηθήσει για την ανάπτυξη Διαδικτυακών εφαρμογών με καλές επιδόσεις, που πλησιάζουν τις επιδόσεις των αυτόνομων προγραμμάτων.
- Το πρότυπο επιτρέπει, με τη βοήθεια των κατάλληλων εργαλείων, την αυτόματη μετατροπή έτοιμου κώδικα άλλων γλωσσών προγραμματισμού, σε μορφή WebAssembly.
- Η βελτίωση των επιδόσεων εξαρτάται σε μεγάλο βαθμό από το είδος των εφαρμογών και των υπολογισμών που πραγματοποιούν.

Ευχαριστώ!