Defensive JS in JSCert Initial steps

Petar Maksimović

Inria Rennes - Bretagne Atlantique

ANR AJACS Meeting Paris, 27/11/2015

◆□ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ <

Current State of JSCert

<□ ▶ < @ ▶ < E ▶ < E ▶ E の Q @ 2/28

Coverage of JSCert @ POPL'14

ecina	ecma	ecma
	All Annual and Annual Annua	111.6 The Beauty Operation TTP - Lot Reset Dis Constantion TTP - Lot Reset Dis Constantion
rieria ~~	11 Tel conserve Asservation Net. 11 Tel Francisco de Conservation Specification Figure de	1133 Partice Render
	A STATE AND ADDRESS OF A DREAM AND ADDRESS	1122 Participation
Apriles all	a the Aventry Charter (from)	1111 Party research speaker
and and a second se	All the party between all between the set of	TTR Units The store
and a share at	The second to have been a second seco	1111 The science Generative
anies 1	All and a second s	114.3 The researd Operator
anguage therefore a	A DE	1112 Parts Decement Opening
Digits Boot Turber antibilitations	and provide the second s	1147 Dary - Openin
ante pré philipie é	A DE DESERVICIÓN DE	1114 Blacks RCF Connector (-)
artistic and Losing Comman.	A STATE Description of the Area and Ar	114 Bullyholds Operation
Extend Area Experiments 7	a function of futing a	1122 Reving to Constra
The Burners Borney Departure of the Burners of the	Al Martin C	1113 Appring the Liberatory
The Additionary Statement of St	ALL Transport of the Being Type	TILL The Administration (a)
igentia fameritara	11 Sector Street Street	112.2 Approve the Addison Specifics to Randows
pte feel	17 Salaria (reges a la rege)	FLEF The Lat Ball Spectra (+)
nose formal Control Danalers In	A ST Trades Andrew State Type	1173 The Designed Hight Book Special (111)
na teminan is	1.1 Decidentaria	TTELT The case that features (11)
and the second s	11 Technology Appellan Bartina Ba	1123 The South Part Newson (1) 1123 The Cese Ren er wood Chamber (11)
And the factors and startifiers.	The Associate Sale and Frencher Saleshing	FIER The Sealer Barranese Convert (in)
and and a second s	The second	TIES The Name of Gentlet
Lui Janit	The Designed Reads	The Assertic Sector
Scout Linear	1012 The Date Sciences	T121 The figure Openion (++)
Trapie Comasse Likeus	ALL MARKA REALES	1123 The Andread Facility Comparison Reportion
Rute of Automatic Restance Pagetan	Here Andread and Andread Andrea	TILL The loss has unique feasible (to)
Examples of Automatic Servicelus Insertion 27	And Second and Add	1112 Brury Steve Constru-
in product law	No. Second Subscrepture	1112 Constant Connect (1 - 1
te Bonter Type	a layered and a second se	1113 Bargerand Garates
to turning Type	WARE WE CONTRACT OF THE OWNER	1111 Control Constant ()
Property All States	The second se	12 Reports
a have been a second of the	The second of th	THE TANKS BARRIER
Handood 201	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	El fores Hermatives 2011
cma	ecma	
cma	ecma	
cma		
CCMA Normal Networks	Cecma	·····
		r
	CONTRACTORY AND A CONTRAC	Not covered
		Not covered
	CONTRACTORY CONTRACTO	Not covered
	CONTRACTOR OF A CONTRACTOR OF	Not covered
	CONSTRUCTION Additional and a second secon	Not covered
Cma the second	COMPACT STATEMENT STA	Not covered
		Not covered
CIDA		Not covered
	Control and an analysis Control an	Not covered
	COMMENTATION C	Not covered Fully Covered
	Control and a second seco	Not covered Fully Covered Specified
	CONTRACTORS	Not covered Fully Covered Specified
Cina Markan Mark	Control and an	Not covered Fully Covered Specified
	ECCCCC Alternative Statement A	Not covered Fully Covered Specified
	CONTRACTORS	Not covered Fully Covered Specified
		Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
	COMPAREMENT Compareme	Not covered Fully Covered Specified For-in
	ECCENT	Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
	EVERONE A series of the serie	Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
	Control of the second s	Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in
		Not covered Fully Covered Specified For-in

・ロト <
一 ト <
一 ト <
三 ト <
三 ト <
三 ト <
三 ・
、
三 ・
の へ の 3/28
</p>

Coverage of JSCert @ POPL'14

Main functionalities (Ch 8-14) dominantly implemented

- For-in broken, ergo not implemented
- Functionalities depending on for-in not implemented
- Libraries (Function, Array, String... (Ch 15)) not formalised

▲□▶▲□▶▲□▶▲□▶ □ のへで 4/28

- Functionalities depending on Ch 15 not formalised
- Array initialiser not formalised

How to improve JSCert?

- Better Coverage of the Standard Libraries
 We need Ch 15 to fully test Ch 8–14 using Test262
- Better Testing Infrastructure

We need to increase our confidence in the test results

- Separate core from non-core library functionality
 - Core functionality: everything the underlying implementation must provide
 - Non-Core functionality: everything that can be implemented in JavaScript on top of core functionality
- Extend JSCert, JSRef, proof with core library functionality
- Implement (or borrow an implementation of) non-core functionality in JavaScript and load it into the initial heap

Coverage of JSCert

ecma ecma ecma Marchan Marchanger Grander Company of Stationer Company of Stationer Research Kinth Anchaiten Clante Kulturin Kulturin Kunturin Linaran Kunturin Linaran Kunturin Linaran Kunturin Linaran Kunturin Linaran Kulturin Kunturin Kunturin Kuturin Kunturin Kunturin ecma ecma Not covered Fully Covered @ POPL'14 Specified Construction Calified as a 74 For-in the disk of the [] Parallele Reports The Function Community Called as a Narother Fully Covered @ CAV'15 Alterated Danse Longs V8

Coverage of JSCert - Detailed Results

15.1 The Global Object 103 Global Object 15.1 Yalue Properties of the Global Object 104 104 15.1.2 Function Properties of the Global Object 104 104 15.1.3 URI Handling Function Properties 106 104 29/32 15.1.4 Constructor Properties of the Global Object 104 105 29/32 15.2 Object Objects 111 111 29/32 0bject Object Objects 111 15.2 Object Object Constructor Called as a Function 111 111 111 111 15.2 Object Constructor Called as a Function 111 111 111 112 15.2.3 Properties of the Object Constructor 112 111 111 111 15.2.5 Properties of Object Constructor 1111 111 111 1	15 Standard Built-In ECMAScript Objects 102	
15.1.1 Value Properties of the Global Object 100 15.1.2 Function Properties of the Global Object 100 15.1.3 URI Handling Function Properties of the Global Object 100 15.1.4 Constructor Properties of the Global Object 100 15.1.5 Other Properties of the Global Object 100 15.1.5 Other Properties of the Global Object 100 15.2.1 The Object Constructor Called as a Function 111 15.2 Properties of the Object Prototype Object 111 15.3 Properties of Object Instances 117 15.3 Function Objects 117 15.3 Function Objects 117 15.3 The Function Constructor Called as a Function 117 15.3 The Function Constructor Called as a Function 117 15.3 The Function Constructor Called as a Function 117 15.3 Properties of the Function Prototype Object 118 15.4 Properties of the Function Restructor 118 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Prototype Object 123 15.4 Properties of the Array Prototype Object 123 <td>15.1 The Global Object</td> <td>Clahal Ohiaat</td>	15.1 The Global Object	Clahal Ohiaat
15.12 Function Properties of the Global Object 104 15.13 URI Handling Function Properties of the Global Object 106 15.14 Constructor Properties of the Global Object 106 15.15 Other Properties of the Global Object 101 15.2 Object Constructor Called as a Function 111 15.2 The Object Constructor 112 15.2.1 The Object Constructor 112 15.2.3 Properties of the Object Constructor 112 15.2.4 Properties of the Object Constructor 112 15.2.5 Properties of Object Constructor 112 15.4 Properties of the Constructor Called as a Function 117 15.3 Function Constructor Called as a Function 117 15.3.4 Properties of the Function Constructor 117 15.3.5 Properties of the Function Prototype Object 118 15.4 Array Objects 121 15.4 Array Constructor 122 15.4.3 Properties of the Array Constructor 122 15.4.4 Properties of the Array Prototype Object 123 15.4.5 Prop	15.1.1 Value Properties of the Global Object	Global Object
15.13 URI Handling Function Properties 106 29/32 15.14 Constructor Properties of the Global Object 106 29/32 15.15 Other Properties of the Global Object 111 Object Object Object 111 15.2 The Object Constructor Called as a Function 111 Object Object 0bject Object 15.2 The Object Constructor Called as a Function 111 111 0bject Object 15.3 Properties of the Object Prototype Object 112 19/24 19/24 15.3 Properties of Object Instances 117 Function Objects 113 15.3 The Function Constructor Called as a Function 117 Function Objects 15.3.1 The Function Constructor Called as a Function 117 Function Objects 15.3.3 Properties of the Function Prototype Object 118 13/15 15.4 Properties of Unction Instances 121 13/15 15.4 Properties of the Array Constructor 122 13/29 15.4.1 The Array Constructor 122 11/29 11/29 15.4.2 Properties of Array Prototype Object 123 <	15.1.2 Function Properties of the Global Object	
15.14 Constructor Properties of the Global Object. 110 29/32 15.15 Other Properties of the Global Object. 111 Object Objects 15.2 Object Constructor Called as a Function 111 0bject Object 15.2.7 The Object Constructor 111 0bject Object 15.2.8 Properties of the Object Constructor 112 111 15.2.9 The Object Constructor 112 111 15.2.9 Properties of the Object Constructor 112 111 15.2.9 Properties of the Object Constructor 112 111 15.2.9 Properties of the Object Constructor 112 111 15.3 Function Constructor Called as a Function 117 Function Objects 15.3.1 The Function Constructor 117 Function Objects 113/15 15.4.7 Properties of the Function Prototype Object 118 13/15 15.4.4 Properties of the Array Prototype Object 122 15.4.3 Properties of the Array Prototype Object 123 15.4.4 Properties of the Array Prototype Object 123 15.4.5 Properties	15.1.3 URI Handling Function Properties	
15.1.5 Other Properties of the Global Object 111 Lord L 152 Object Objects 111 Object Object Object Object Instances 0 153.2 The Object Constructor Called as a Function 111 0 0 153.2 The Object Constructor 112 19/24 19/24 153.4 Properties of the Object Instances 117 117 Function Objects 153.1 The Function Constructor Called as a Function 117 117 Function Objects 153.3 Properties of the Function Constructor 117 118 13/15 13/15 153.4 Properties of the Function Constructor 118 13/15 13/15 13/15 153.4 Properties of the Function Constructor 118 13/15 13/15 13/15 154.4 Properties of Function Constructor 122 13/15 13/15 13/15 154.4 Properties of the Array Constructor 122 11/29 11/29 11/29 154.4 Properties of Array Prototype Object 123 140 18/29 18/29	15.1.4 Constructor Properties of the Global Object	29/32
152 Object Objects 111 Object Object 152.1 The Object Constructor Called as a Function 111 Object Object 152.2 The Object Constructor 112 111 111 152.3 Properties of the Object Constructor 112 19/24 152.4 Properties of the Object Constructor 111 19/24 152.5 Properties of the Object 111 19/24 153.4 Properties of the Function Constructor Called as a Function 117 111 153.3 Properties of the Function Constructor 117 13/15 13/15 153.4 Properties of the Function Constructor 118 13/15 13/15 153.4 Properties of the Function Prototype Object 122 13/15 13/15 154.4 Properties of the Array Constructor 122 11/29 11/29 154.4 Properties of the Array Prototype Object 123 11/29 18/29 154.5 Properties of Array Instances 140 18/29 18/29	15.1.5 Other Properties of the Global Object	20/02
15.2 Object Objects 0bject Constructor Called as a Function 0bject Object Constructor 15.2.1 The Object Constructor 111 0bject Object Constructor 15.2.2 The Object Constructor 112 114 15.2.3 Properties of the Object Constructor 112 114 15.2.4 Properties of the Object Constructor 117 119/24 15.3 Properties of Dbject Instances 117 117 15.3 The Function Constructor Called as a Function 117 113/15 15.3.1 The Function Constructor Constructor 117 113/15 15.3.2 The Function Constructor 117 113/15 15.3.4 Properties of the Function Constructor 117 113/15 15.4 Properties of the Function Constructor 117 113/15 15.4 Properties of the Array Constructor 122 113/15 15.4 Properties of the Array Constructor 122 15.4.3 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Prototype Object 123 15.4.5 Properties of Array Prototype Object <		
152.1 The Object Constructor Called as a Function 111 Object Constructor 152.2 The Object Constructor 112 152.3 Properties of the Object Prototype Object 112 15.4 Properties of the Object Prototype Object 115 15.5 Properties of Object Instances 117 15.3 Properties of Ubject Instances 117 15.3 Properties of the Function Constructor Called as a Function 117 15.3 Properties of the Function Constructor 116 15.3 Properties of the Function Constructor 116 15.4 Properties of the Function Instances 121 15.4 Properties of the Function Constructor 118 15.4 Properties of the Function Constructor 118 15.4 Properties of the Function Constructor 122 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Prototype Object 123 15.4 Properties of the Array Prototy	15.2 Object Objects	Object Object
15.22 The Object Constructor 112 15.23 Properties of the Object Constructor 112 15.24 Properties of the Object Constructor 112 15.25 Properties of the Object Constructor 112 15.26 Properties of Object Instances 117 15.3 Properties of Mobject Constructor Called as a Function 117 15.3 The Function Constructor Called as a Function 117 15.3.1 The Function Constructor Called as a Function 118 15.3.2 The Function Constructor 118 15.3.4 Properties of the Function Prototype Object 118 15.4 Array Objects 112 15.4 Array Constructor 122 15.4.1 The Array Constructor 122 15.4.2 The Array Prototype Object 123 15.4.4 Properties of Haray Prototype Object 123 15.4.5 Properties of Array Prototype Object	15.2.1 The Object Constructor Called as a Function	Object Object
15.2.3 Properties of the Object Prototype Object. 112 19/24 15.2.4 Properties of Object Instances 117 117 15.3.5 Properties of Object Instances 117 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 117 117 15.3.2 The Function Constructor Called as a Function 117 117 117 15.3.2 The Function Constructor Called as a Function 117 117 13/15 15.3.4 Properties of the Function Prototype Object 118 13/15 13/15 15.4 Properties of the Function Constructor 122 13/15 Array Objects 15.4 Array Objects 122 13/15 11/29 11/29 15.4.3 Properties of the Array Prototype Object 123 11/29 11/29 15.4.4 Properties of Array Instances 140 18/29 18/29	15.2.2 The Object Constructor	
15.2.4 Properties of the Object Prototype Object. 117 15.2.5 Properties of Object Instances 117 15.3 Function Objects 117 15.3 Function Objects 117 15.3 Function Constructor Called as a Function 117 15.3 Function Constructor Called as a Function 117 15.3 Function Objects 118 15.3 Function Constructor 118 15.3 Properties of the Function Prototype Object 118 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Prototype Object 123 15.4 Properties of the Array Prototype Object 123 15.4 Properties of Array Instances 140 18/29 18/29	15.2.3 Properties of the Object Constructor	10/2/
152.5 Properties of Object Instances 117 15.3 Function Objects 117 15.3 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor Called as a Function 117 15.3.3 Properties of the Function Constructor 118 15.3.4 Properties of the Function Constructor 118 15.3.5 Properties of the Function Instances 121 15.4 Array Objects 122 15.4 Array Objecties of the Array Constructor 122 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Prototype Object 123 15.4 Properties of the Array Constructor 122 15.4 Properties of the Array Prototype Object 123 15.4.5 Properties of Array Instances 140 18/29 18/29	15.2.4 Properties of the Object Prototype Object	19/24
15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 118 15.3.4 Properties of the Function Prototype Object 118 15.4 Array Objects 121 15.4 Array Objects 122 15.4.1 The Array Constructor 122 15.4.2 The Array Prototype Object 123 15.4.4 Properties of Haray Prototype Object 123 15.4.5 Properties of Array Prototype Object 123 15.4.4 Properties of Array Prototype Object 123 15.4.5 Properties of Array Instances 140	15.2.5 Properties of Object Instances	
15.3 Function Objects 117 Function Objects 15.3.1 The Function Constructor Called as a Function 117 Function Objects 15.3.2 The Function Constructor Called as a Function 117 Function Objects 15.3.3 Properties of the Function Constructor 118 13/15 15.3.4 Properties of the Function Instances 121 13/15 15.4 Array Objects 122 15/4 Array Objects 15.4 Tray Constructor 122 11/29 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Instances 140		
15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.4 Properties of the Function Constructor 118 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor Called as a Function 122 15.4.2 The Array Constructor 122 15.4.3 Properties of the Array Prototype Object 123 15.4.4 Properties of the Array Prototype Object 123 15.4.5 Properties of Array Instances 140		
15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 118 15.3.4 Properties of the Function Constructor 118 15.3.5 Properties of Unction Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor 122 15.4.2 The Array Constructor 122 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Instances 140 15.4.5 Properties of Array Instances 140	15.3 Function Objects	Function Objects
15.3.3 Properties of the Function Constructor 118 13/15 15.3.4 Properties of the Function Instances 121 13/15 15.4.5 Properties of Function Instances 121 121 15.4.7 Properties of the Array Constructor 122 121 15.4.7 The Array Constructor 122 122 15.4.8 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Prototype Object 123 15.4.5 Properties of Array Instances 140	15.3 Function Objects	Function Objects
15.3.4 Properties of the Function Instances 13 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor 122 15.4.2 The Array Constructor 122 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Instances 140	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.2 The Function Constructor 117	Function Objects
153.5 Properties of Function Instances 121 154 Array Objects 122 15.4 Array Constructor Called as a Function 122 15.4.2 The Array Constructor 122 15.4.3 Properties of the Array Prototype Object 123 15.4.4 Properties of Array Prototype Object 123 15.4.5 Properties of Array Prototype Object 123 15.4.5 Properties of Array Instances 140	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117	Function Objects
15.4 Array Objects 122 Array Objects 15.4.1 The Array Constructor Called as a Function 122 Array Objects 15.4.2 The Array Constructor 122 122 15.4.3 Properties of the Array Prototype Object 123 11/29 15.4.5 Properties of Array Instances 140 18/29	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.4 Properties of the Function Constructor 118 15.3.4 Properties of the Function Prototype Object 118	Function Objects
15.4 Array Objects 122 Array Objects 15.4.1 The Array Constructor Called as a Function 122 122 122 15.4.2 The Array Constructor 122 123 134 The Array Constructor 122 11/29 15.4.3 Properties of the Array Prototype Object 123 11/29 18/29 15.4.5 Properties of Array Instances 140 18/29 18/29	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 118 15.3.3 Properties of the Function Prototype Object 118 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of the Function Instances 128	Function Objects
15.4.1 The Array Constructor 122 111/29 15.4.2 The Array Constructor 122 11/29 15.4.3 Properties of the Array Prostryce Object. 123 11/29 15.4.4 Properties of the Array Prostryce Object. 123 11/29 15.4.5 Properties of Array Instances 140 18/29	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Prototype Object 118 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of the Function Instances 121	Function Objects
15.4.2 The Array Constructor 122 15.4.3 Properties of the Array Youtotype Object 123 15.4.4 Properties of Array Instances 140 15.4.5 Properties of Array Instances 140	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.4 Properties of the Function Constructor 118 15.3.4 Properties of Function Prototype Object 118 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122	Function Objects
15.4.3 Properties of the Array Constructor 123 11/29 15.4.4 Properties of the Array Prototype Object 123 11/29 15.4.5 Properties of Array Instances 140 18/29	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of the Function Prototype Object 118 15.3.4 Properties of Function Instances 121 15.4 Array Objects 122	Function Objects 13/15 Array Objects
15.4.4 Properties of the Array Prototype Object	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor Called as a Function 122 15.4.2 The Array Constructor 122	Function Objects 13/15 Array Objects
15.4.5 Properties of Array Instances	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 118 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of the Function Prototype Object 118 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor 122 15.4.2 The Array Constructor 122 15.4.3 Properties of Function Instances 121	Function Objects 13/15 Array Objects 11/29
18/29	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.4 Properties of the Function Constructor 118 15.3.5 Properties of Function Prototype Object 118 15.4.7 The Array Constructor Called as a Function 122 15.4.1 The Array Constructor 122 15.4.2 The Array Constructor 122 15.4.4 Properties of the Array Porotype Object 123 15.4.4 Properties of the Array Prototype Object 123	Function Objects 13/15 Array Objects 11/29
10/23	15.3 Function Objects 117 15.3.1 The Function Constructor Called as a Function 117 15.3.2 The Function Constructor 117 15.3.3 Properties of the Function Constructor 117 15.3.3 Properties of the Function Constructor 118 15.3.4 Properties of the Function Prototype Object 118 15.3.5 Properties of Function Instances 121 15.4 Array Objects 122 15.4.1 The Array Constructor 122 15.4.2 The Array Constructor 122 15.4.3 Properties of the Array Constructor 123 15.4.4 Properties of the Array Constructor 123 15.4.3 Properties of the Array Constructor 123 15.4.4 Properties of the Array Constructor 124 15.4.5 Properties of the Array Constructor 124 15.4.5 Properties of the Array Postotype Object 124 15.4.5 Properties of the Array Postotype Object 124 15.4.5 Properties of the Array Postotype Object 124	Function Objects 13/15 Array Objects 11/29

V8

Fully Covered @ CAV'15

Coverage of JSCert - Detailed Results

15 8	Standard Built-in ECMAScript Objects	102
45.5	Alter Alterta	
15.5	String Objects	141
15.5.1	The String Constructor Called as a Function	141
15.5.2	The String Constructor	142
15.5.3	Properties of the String Constructor	142
15.5.4	Properties of the String Prototype Object	142
15.5.5	Properties of String Instances	151

15.6	Boolean Objects	52
15.6.1	The Boolean Constructor Called as a Function1	52
15.6.2	The Boolean Constructor	52
15.6.3	Properties of the Boolean Constructor	53
15.6.4	Properties of the Boolean Prototype Object.	53
15.6.5	Properties of Boolean Instances	53

15.7	Number Objects
15.7.1	The Number Constructor Called as a Function
15.7.2	The Number Constructor154
15.7.3	Properties of the Number Constructor154
15.7.4	Properties of the Number Prototype Object
15.7.5	Properties of Number Instances

15.8	The Math Object	159
15.8.1	Value Properties of the Math Object	159
15.8.2	Function Properties of the Math Object	160

String Objects 5/26

2/26

Boolean Objects

7/7

Number Objects

12/16

Math Object

Fully Covered @ CAV'15

Specified

Coverage of JSCert - Detailed Results

15	Standard Built-in ECMAScript Objects 102		
15.9	Date Objects	Date	
15.9	1 Overview of Date Objects and Definitions of Abstract Operators 165	.	
15.9	2 The Date Constructor Called as a Function 170	Objects	
15.9	The Date Constructor 170		
15.9	A Properties of the Date Constructor 171		
15.9	5 Properties of the Date Prototype Object 172		
15.9	6 Properties of Date Instances 180		
10.0		Dogular	
	2 Des Fuer (Desudes Fuerencies) Objects	negulai	
10.1	180	Evorose	sione
15.1	100	Lybies	510115
15.1	0.2 Pattern Semantics		
15.1	194		
15.1	194		
15.1	U.5 Properties of the RegExp Constructor		
15.1	0.5 Properties of the HegExp Prototype Object		
15.1	197	-	
		Error Or	ojects
15.1	1 Error Objects		-
15.1	1.1 The Error Constructor Called as a Function	12/25	
15.1	1.2 The Error Constructor	12/25	
15.1	1.3 Properties of the Error Constructor		
15.1	1.4 Properties of the Error Prototype Object	11/25	
15.1	1.5 Properties of Error Instances		
15.1	1.6 Native Error Types Used in This Standard		
15.1	1.7 NativeError Object Structure		
15.1	2 The JSON Object		
15.1	2.1 The JSON Grammar		
15.1	2.2 parse (text [, reviver])		

- 15.12.3 stringify (value [, replacer [, space]]).....
- Errors

Fully Covered @ CAV'15



208

Coverage of JSCert - Array Library

15.4	Array Objects	122
15.4.1	The Array Constructor Called as a Function	122
15.4.2	The Array Constructor	122
15.4.3	Properties of the Array Constructor	123
15.4.4	Properties of the Array Prototype Object	123
15.4.5	Properties of Array Instances	140

◆□ ▶ < □ ▶ < ■ ▶ < ■ ▶ ■ のへで 10/28</p>

Fully Covered @ CAV'15 - 11 sections out of 29



JSRef Testing Results: Array Library

	Chs 8-14			Ar	ray Lib	rary
	Pass	Fail	Abort	Pass	Fail	Abort
POPL'14 paper	1796	404	582	139	873	1307
POPL'14 talk	1851	392	539	149	864	1306
CAV'15 paper	2437	129	216	180	1204	935
+V8 Array	2440	126	216	1309	59	951
CAV'15 talk	2506	47	229	267	1956	69
+V8 Array	2510	43	229	2170	12	111

Evaluation

- Implementation of new features uncovered bugs in previously unused and unverified code.
- Changing the parser forced re-evaluation of parsing failures

Defensive JavaScript

<□ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □

What is DJS and why was it created?

- Typed subset of JavaScript
- Authors: K. Bhargavan, A. Delignat-Lavaud, S. Maffeis
- Motivation: Guarantee that program functionalities cannot be tampered with, even in a malicious environment

▲□▶▲□▶▲□▶▲□▶ □ のへで 13/28

 Applications: Crypto libraries, single sign-on widgets, bookmarklets

What is meant by "defensiveness"?

Isolate security-critical code from the environment:

< □ ▶ < □ ▶ < 三 ▶ < 三 ▶ E りへで 14/28

- Use function closures/wrappers
- Do not call external function explicitly
- Prevent triggering of coercions
- Prevent prototype lookups

How is this achieved?

Using a static type system:

- Static scopes
 - Limited occurrences of variable declarations,
 - Strong scoping restriction on with
- Statically typed objects, functions, arrays
 - No out-of-bounds, not extensible
- Coercion-free operations
- Disjoint heaps
 - No heap references allowed to be imported or exported, only string → string

▲□▶▲□▶▲≣▶▲≣▶ ≣ のへで 15/28

Syntactic categories

- Literals (bool, number, string, object, array)
- Most unary and binary operators
- LHS-expressions (severely limiting dynamic access)
 - Properties only via e.x
 - Arrays and strings indexed only within bounds
- Expressions
 - Assignments; unary, binary operations
 - Fully applied function and method applications

< □ ▶ < □ ▶ < 三 ▶ < 三 ▶ E り < ○ 16/28

- Statements
 - If-then-else, while, sequence
 - with(e)s, all FV(s) are properties of e
 - No variable declarations

Syntactic categories

Functions

- Variable declarations
- Series of statements
- Single return statement

$$f := \texttt{function}(x_1, \dots, x_n) \{$$
$$\texttt{var } y_1 = e_1, \dots y_m = e_m;$$
$$s_1; \dots; s_k; \texttt{return } e \}$$

◆□ ▶ < □ ▶ < ■ ▶ < ■ ▶ < ■ ● ○ Q ○ 17/28</p>

Syntactic categories

Programs

- Wrapper around a single function *f*
- Ensures argument is string
- Calls function, returns result

$$p_f := (function () \{ \\ var_- = f; \\ return function(x) \{ if (typeof(x) == "string") \\ return _(x); \} \})();$$

- Wrapping—no leaking of source code of *f*
- Argument type check—no import of external heap refs

Types and Environments

Types and Environments

	$\tau ::=$	Types
	number boolean string undefine	ad Base Types
	ρ	Object
	$[\tau]_n$	Array of length $n \ge 0$
	$(\tau_1,\ldots,\tau_n) \to \tau_e$	Function $n \ge 0$
	$(\tau_1,\ldots,\tau_n)[\rho] \to \tau_r$	Method on object of type ρ $(n \ge 0)$
	$\rho ::= \{x_1 : \tau_1, \dots, x_n : \tau_n\}$	Object Type $(n \ge 0)$
	$\kappa ::= f \mid w$	Scope frame kind: function or with
	$\Gamma ::= \varepsilon \mid \Gamma, [\rho]_{\kappa}$	Typing Environment
L		

▲□▶▲舂▶▲≧▶▲≧▶ ≧ のへで 19/28

Illustration of the typing rules

$$\begin{split} & \operatorname{StrCast} \underbrace{-\frac{\Gamma \vdash e : \operatorname{number}}{\Gamma \vdash e + "" : \operatorname{string}}}_{\Gamma \vdash e_1, \ldots, e_n] : [\tau]_n} \quad \operatorname{ConstantIndex} \underbrace{-\frac{\Gamma \vdash e : [\tau]_m \quad m > \eta \ge 0}{\Gamma \vdash e[\eta] : \tau}}_{VarLocal} \underbrace{-\frac{\Phi(x) = \tau}{\Gamma, [\Phi]_\kappa \vdash x : \tau}}_{\Gamma, [\Phi]_\kappa \vdash x : \tau} \quad \operatorname{VarFunctionScope} \underbrace{-\frac{x \not\in \operatorname{dom}(\Phi) \quad \Gamma \vdash x : \tau}{\Gamma, [\Phi]_f \vdash x : \tau}}_{\Gamma, [\Phi]_f \vdash x : \tau} \\ & \operatorname{Petric}_{\Gamma, [\rho_m]_f \vdash d_k : \mu_k \quad k \in [1..m]}_{\Gamma, [\rho_m]_f \vdash e_s : \operatorname{undefined} \quad \Gamma, [\rho_m]_f \vdash e_r : \tau_r} \end{split}$$
FunctionDef
$$\underbrace{-\frac{\Gamma, [\rho_m]_r \vdash s : \operatorname{undefined} \quad \Gamma, [\rho_m]_f \vdash e_r : \tau_r}{\Gamma \vdash \operatorname{function}(\tilde{x})\{\operatorname{var} y_1 = d_1, \ldots, y_m = d_m; \ s \ ; \ \operatorname{return} \ e_r\} : \tilde{\tau} \to \tau_r} \end{split}$$

◆□ ▶ < 圕 ▶ < Ξ ▶ < Ξ ▶ Ξ の Q ? 20/28</p>

Key Properties of DJS

Independence: p_f preserves the independence of f if any two sequences of calls with the same arguments to the result of p_f , interleaved with arbitrary JS code, return the same sequences of return values, provided no call triggered an exception.

Encapsulation: p_f encapsulates f over \mathcal{D} if no JS program that runs p_f can distinguish between running p_f and $p_{f'}$ without calling the returned wrapped functions. Moreover, for any tuple $\tilde{v} \in \mathcal{D}$, heaps resulting from $p_f(\tilde{v})$ and $f(\tilde{v})$ are equivalent.

Defensiveness: If $\vdash f : string \rightarrow string$, then the DJS program p_f encapsulates f over strings and preserves its independence.

What has been done so far in Coq?

- 1. Complete Syntax
- 2. Complete Type System
- 3. Mapping $DJS \rightarrow JSCert$
- 4. Predicate describing allowed JSCert terms

<□ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

5. Proof that (3) and (4) are equivalent

Defined size for DJS types, DJS terms, JSCert terms

- All proofs by induction on appropriate size
- Using size, derive structural induction principles

<□ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

- Rules of variable length (object, array, function)
 - Using Forall from TLC
 - Pretty-big-step style rules
 - PBS for functions, Forall for the rest

```
| typing_Array : forall G t le,
    length le > 0 ->
    Forall (fun e => G |- e :: t) le ->
    G |- (djs_a le) :: [[t, length le]]
```

- ► Translation DJS → JSCert (toJSC)
 - Mapping of unary and binary operators
 - Follow descriptions from the syntax

```
(* Constant Array Index *)
| djs_cai ε n → expr_access (toJSC_ε ε)
                                  (expr literal (literal number (JsNumber.of int n)))
(* Functions *)
with toJSC o o :=
match & with
 | dis φ main lx τργ s ε ⇒
       expr function
          None (* Functions have no names *)
          lx (* List of variables
          (* Function body *)
          (funcbody intro
             (prog intro
                true (* Body is always strict *)
                   (* Variables *) element stat (stat var decl
                                                       ((fix var decl τρy :=
                                                            match toy with
                                                              I nil ⇒ nil
                                                              (v, \delta \varepsilon) :: \tau \rho v \Rightarrow (v, Some (to JSC \delta \varepsilon \delta \varepsilon)) :: var decl \tau \rho v
                                                            end) tov)) ::
                   (* Statement *) element stat (toJSC stat s) ::
                               *) element stat (stat return (Some (toJSC ε ε))) :: nil
                   (* Return
             "function() {}"
end
```

- Allowed terms in JSCert (DJS_allowed_term)
 - Mapping of operators
 - Follow descriptions from the syntax

```
(* Functions do not have names, programs always in strict mode *)
| expr function None lx (funcbody intro (prog intro true lel) msg) =>
 msg = "function() {}" /\
 match lel with
  (* first statement is a variable declaration *)
   | element stat (stat var decl τρy) :: lel =>
    ((fix DJS allowed lie τρv :=
        match tpy with
          | nil => True
          | (, Some je) :: \tau py => DJS allowed expr je /\ DJS allowed lje \tau py
           => False
         end) \tau \rho \gamma) /\
    match lel with
       (* functions = statement, then return *)
       | element stat s :: element stat (stat return (Some je)) :: nil =>
           DJS allowed stat s /\ DJS allowed expr je /\ not a function je
       => False
    end
  _ => False
  end
```

Relationship between toJSC and DJS_allowed_term
 T - DJS term, t - JSCert term

 $\forall t, DJS_allowed_term t \leftrightarrow (\exists T, toJSC T = t)$

<□ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

What are the next steps?

Compatibility between Γ in DJS and (S, C) in JSCert

<□ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > < □ > <

- Type safety
- Properties of the heap
 - Equivalence of heaps
 - Separation
- Additional information within JSCert
 - Traces?
- Semantics of DJS?

Thank you for your attention!

◆□ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ▶ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ∧ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ♪ < □ ∧ < □