

ChakraCore

e o inicio de uma nova
era na comunidade Node.js

Mary Marchini

 @mmarkini

sthima

Mary Marchini

Developers Team Leader @ **Sthima**

Node.js Collaborator

Node.js Diagnostics WG Member



 @mmarkini

 <http://mmarchini.me/>

 oss@mmarchini.me

 <https://github.com/mmarchini/>

O que é ChakraCore?

O que é Node.js?

8.8 Milhões
de instâncias rodando

47k
Estrelas no GitHub

3+ Bilhões
Pacotes baixados
semanalmente no npm

Oito
Arquiteturas suportadas

1.500+
Contribuidores

4800
Pacotes publicados
semanalmente

Como tudo começou...

2008



2008



2008



V8



V8

- V8 não era um interpretador
 - Ele era um compilador
 - Sempre gerava código de máquina
 - Performance decente para uma linguagem dinâmica na época

V8

- V8 não era um interpretador
 - Ele era um compilador
 - Sempre gerava código de máquina
 - Performance decente para uma linguagem dinâmica na época
- Open-source

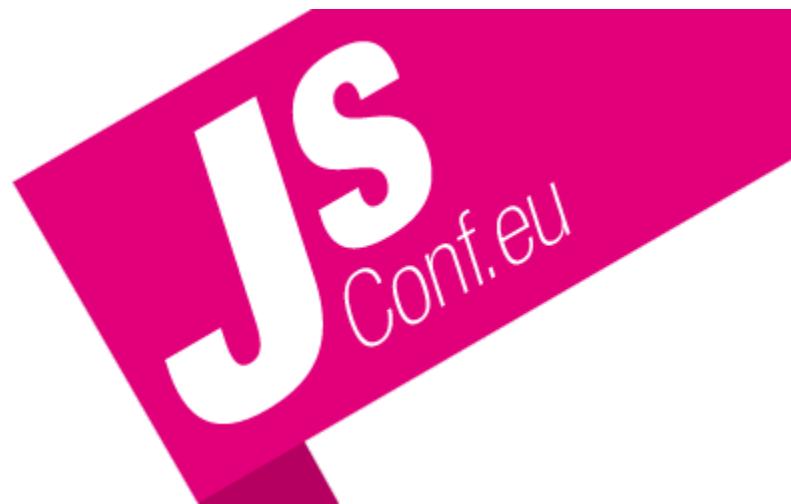
V8

- V8 não era um interpretador
 - Ele era um compilador
 - Sempre gerava código de máquina
 - Performance decente para uma linguagem dinâmica na época
- Open-source
- Embarcável

2009



2009



2009



2009



JSConfEU 2009 - Ryan Dahl apresenta
Node.js pela primeira vez

2009



JSConfEU 2009 - Ryan Dahl apresenta
Node.js pela primeira vez

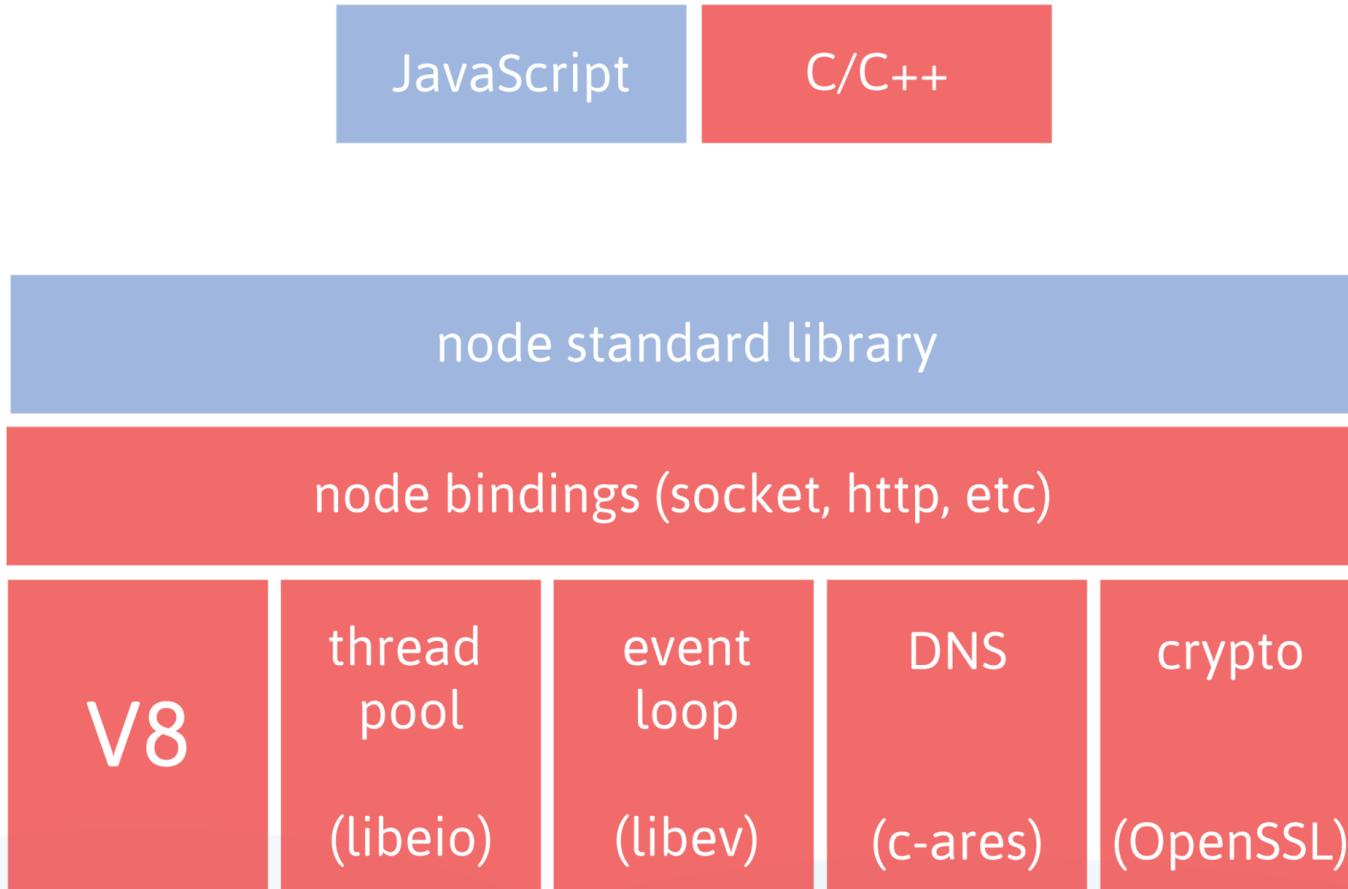
*“ I/O has to be done
differently. We’re doing it
wrong*

Ryan Dahl, JSConfEU 2009

Node.js

- Javascript para servidor
- Criado em cima do Google V8
- Evented, non-blocking I/O. Parecido com EventMachine e Twisted
- Sistema de módulos CommonJS
- 8000 linhas de C/C++. 2000 linhas de Javascript. 14 contribuidores

Arquitetura Node.js (2009)



2009

Preview: npm, the node package manager

8 posts by 5 authors  



Isaac Schlueter



I think node needs a package manager. There are a lot of very useful modules out there, but it's tricky right now to actually use more than one of them together.

Here's a proposal for a very lightweight and simple way to alleviate the situation. I'm calling it npm, and it should be able to install itself fairly soon. :) In fact, calling this a "preview" is a bit disingenuous, as it's rather heavy on "pre" and not so much "view" just yet.

Isaac Schlueter propõe um gerenciador de pacotes para o Node.js

2010

Node.js

- Node.js 0.2.0
- Surgem alguns pacotes
 - Express
 - Socket.io

2010

V8

Novo compilador: Crankshaft

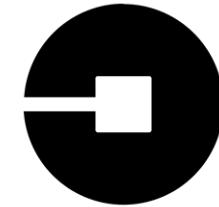
2011



1.0 Released

2011

Adoção do Node.js



UBER

2012

Node.js

- Node.js 0.8.0 (estável)
- Mudança na liderança
 - Ryan Dahl -> Isaac
- Node.js começa a focar no seu ecossistema

2013

Node.js

- Ghost Blog Platform
- MEAN Stack

2013

Adoção do Node.js

PayPal

Walmart 

ebay

2014

Node.js

- Surge um fork do Node.js

2014



2014

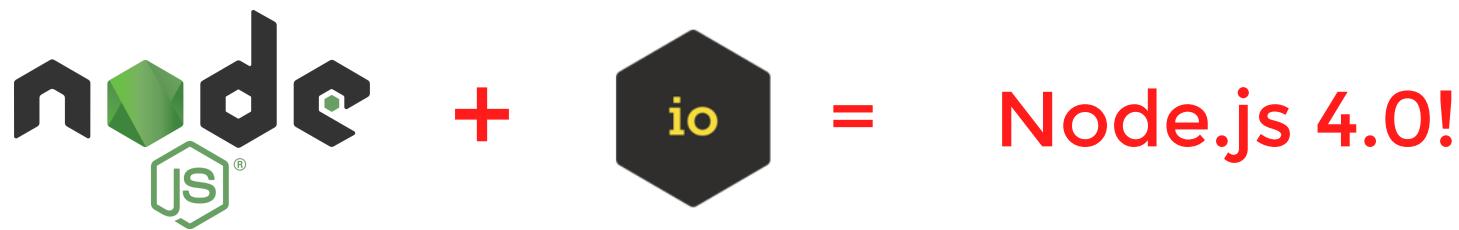
Adoção do Node.js

NETFLIX

2015



2015

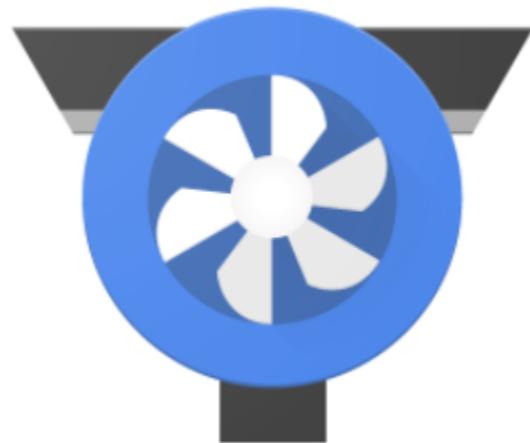


2015

Node.js 4.2.0
LTS

2015

V8 novo compilador:
Turbofan



2016

V8 Lança um interpretador: Ignition



Otimizado para consumo de memória em
dispositivos móveis

2016

Microsoft open sources Edge's Chakra JavaScript engine

by Microsoft + Open Source

January 13, 2016

Application Development

Today, [Microsoft open sourced](#) the key components of the Chakra JavaScript engine that powers Microsoft Edge. The [ChakraCore repository is available today on GitHub](#) and provides a fully supported and open source JavaScript engine, [with the same characteristics](#) as Microsoft Edge's Chakra engine, to [embed in projects, innovate on top of](#) and contribute back to.

SEARCH BLOG



UPCOMING EVENTS

2016

Microsoft open sources Edge's Chakra JavaScript engine

by Microsoft + Open Source

January 13, 2016

Application Development

Today, [Microsoft open sourced](#) the key components of the Chakra JavaScript engine that powers Microsoft Edge. The [ChakraCore repository is available today on GitHub](#) and provides a fully supported and open source JavaScript engine, [with the same characteristics](#) as Microsoft Edge's Chakra engine, to [embed in projects, innovate on top of](#) and contribute back to.

SEARCH BLOG



UPCOMING EVENTS

2016

Enable Node.js to run with Microsoft's ChakraCore engine #4765

[Edit](#)

 **Closed** [kunalspathak](#) wants to merge 9 commits into [nodejs:master](#) from [nodejs:nodejs-chakracore](#) · [Jump to bottom](#)

 Conversation 223

 Commits 9

 Files changed 4,774

+2,776,480 -92 



[kunalspathak](#) (Kunal Pathak) on Jan 19, 2016 

Member



Reviewers



No reviews

Assignees

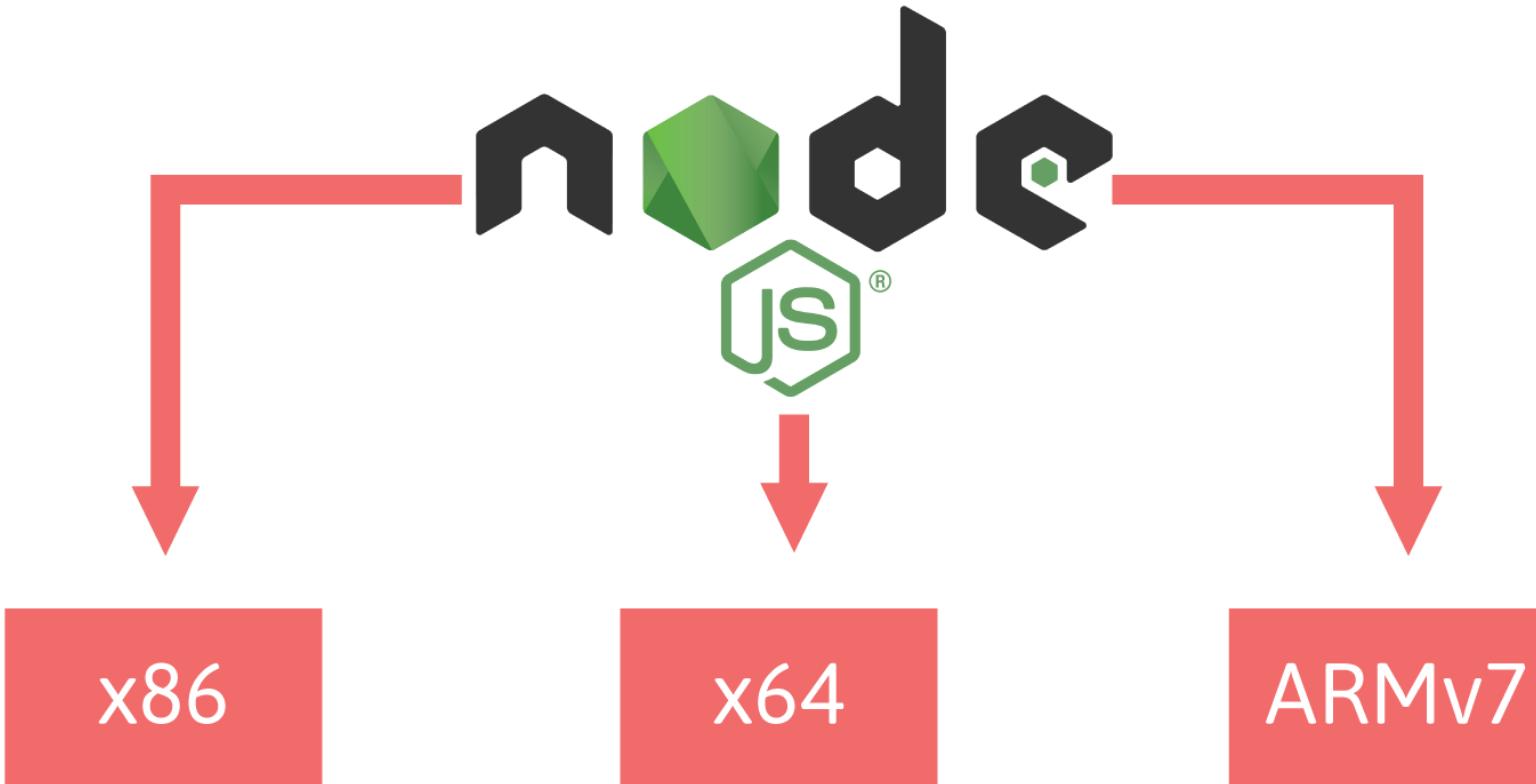


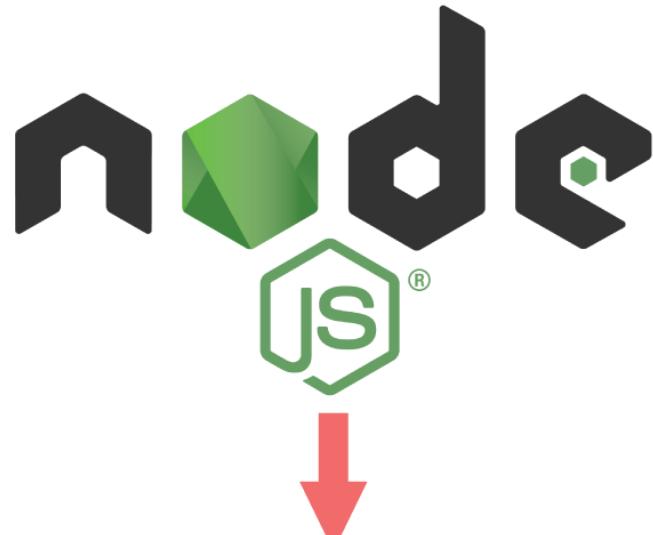
No one—assign yourself

(Note from the CTC (Fishrock123): This thread is expected to garner a lot of attention. Comments that are not productive to discussing the technical aspects may be removed.)

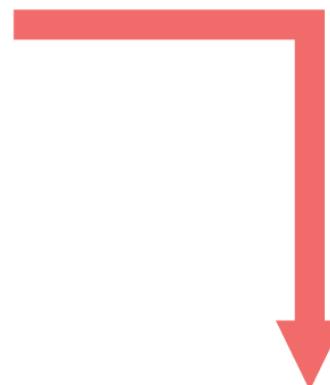
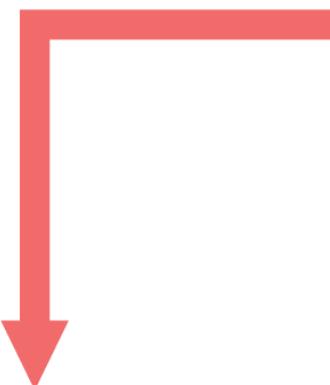
Por que?







ChakraCore



x64

x86

ARM
Thumb-2

2016

 nodejs / node-chakracore 

 Code  Issues 71  Pull requests 4  Wiki  Releases 27 More ▾

Node.js on ChakraCore ✨🐢🚀✨

2016

 [nodejs / node-chakracore](#) Watch 

 [Code](#)  [Issues 71](#)  [Pull requests 4](#)  [Wiki](#)  [Releases 27](#) [More ▾](#)

Node.js on ChakraCore    

API Working Group

The API WG will focus on creating two different specifications. A low-level JavaScript API and an API/ABI compatible native layer.

N-API

- ABI (Application Binary Interface) Stable
- VM neutral

N-API

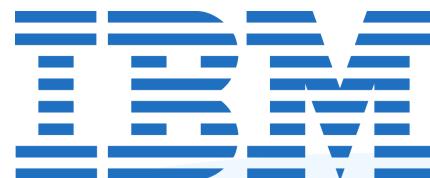
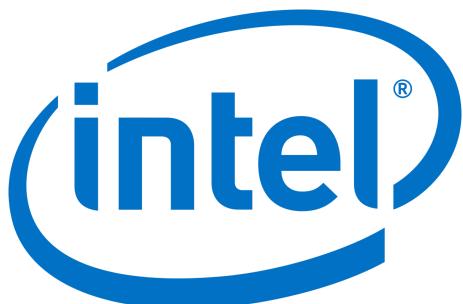
Um módulo nativo (C++) pode ser utilizado:

- Node.js v6.x
- Node.js v8.x
- Node.js v9.x
- Node.js v10.x
- ...
- node-chakracore (v8.x, v9.x, ...)

Sem recompilar!

N-API

Principais Colaboradores



NODESOURCE™

N-API Current Status

- Saiu da versão experimental a alguns meses
 - Com exceção v6.x
- WG está colaborando com criadores de módulos nativos para atualizar para N-API

Time-Travel Debugging



Time-Travel Debugging

- Salva um *trace* da execução do programa

Time-Travel Debugging

- Salva um *trace* da execução do programa
- Um *trace* é uma sequência de passos que o programa realizou durante a sua execução

Time-Travel Debugging

- Salva um *trace* da execução do programa
- Um *trace* é uma sequência de passos que o programa realizou durante a sua execução
- Permite reproduzir esses passos posteriormente, sem efeitos colaterais e com a possibilidade de "*voltar no tempo*"

Time-Travel Debugging

- Salva um *trace* da execução do programa
- Um *trace* é uma sequência de passos que o programa realizou durante a sua execução
- Permite reproduzir esses passos posteriormente, sem efeitos colaterais e com a possibilidade de "*voltar no tempo*"



Time-Travel Debugging

Demo

The screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the following interface elements:

- EXPLORER** sidebar: Shows the project structure with files like `launch.json`, `index.js`, and `launch.json` under `TTD-DEMO`.
- OPEN EDITORS** tab: Shows the currently open files: `launch.json`, `index.js`, and `index.js`.
- TERMINAL** tab: Displays the command `nvs run chakracore-nightly --record index.js` and the timestamp `[12:42:59]`.
- STATUS BAR**: Shows the current file is `index.js — ttd-demo`, line 13, column 1, with 49% completion.

The `index.js` file content is as follows:

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

index.js — ttd-demo

EXPLORER

OPEN EDITORS

- launch.json .vscode
- index.js

TTD-DEMO

- _ttd_log_
- .node-chakracore
- .vscode
- launch.json
- index.js

launched: 1: zsh

PROBLEMS OUTPUT TERMINAL

λ [workspace/tmp/ttd-demo]
→ nvs run chakracore-nightly --record index.js [12:42:59]

Ln 13, Col 1 Spaces: 2 UTF-8 LF JavaScript 49% [|||||] 12:43 PM ☺ 🔔

This screenshot shows a dark-themed instance of Visual Studio Code (VS Code) with the title bar "index.js — ttd-demo". The Explorer sidebar on the left lists files and folders, including "index.js" which is currently open. The main editor area displays a JavaScript file with code that defines two functions, "foo" and "bar", and uses setTimeout to call "bar" with null. The terminal at the bottom has a command history entry: "λ [workspace/tmp/ttd-demo]" followed by "→ nvs run chakracore-nightly --record index.js [12:42:59]". The status bar at the bottom provides information about the current line (Ln 13, Col 1), character count (Spaces: 2), encoding (UTF-8), line separator (LF), language (JavaScript), progress (49% [|||||]), time (12:43 PM), and a smiley face icon.

A screenshot of the Visual Studio Code (VS Code) interface. The workspace is named "tdd-demo".

EXPLORER sidebar:

- OPEN EDITORS:
 - launch.json
 - index.js (highlighted)
- TTD-DEMO:
 - _diagnosticTraces
 - _ttd_log_
 - .node-chakracore
 - .vscode
 - launch.json
 - index.js

CODE EDITOR (index.js tab):

```
function foo(x) {
    return x == null;
}

function bar(x) {
    const shouldCrash = foo(x);
    if (shouldCrash)
        crash();
}

setTimeout(() => { bar(null); });


```

TERMINAL (zsh shell):

```
Recording started (after main module loaded)...
Write error trace to: /Users/mmarchini/workspace/tmp/tdd-demo/_diagnosticTraces/emitOnException_pid48619
/Users/mmarchini/workspace/tmp/tdd-demo/index.js:9
    crash();
^

ReferenceError: 'crash' is not defined
at bar (/Users/mmarchini/workspace/tmp/tdd-demo/index.js:9:5)
at Anonymous function (/Users/mmarchini/workspace/tmp/tdd-demo/index.js:12:20)
at ontimeout (timers.js:426:5)
at tryOnTimeout (timers.js:289:5)
at listOnTimeout (timers.js:252:5)
at processTimers (timers.js:212:3)

λ [workspace/tmp/tdd-demo]
→ █
```

STATUS BAR:

- Ln 13, Col 1 Spaces: 2 UTF-8 LF JavaScript 49% [|||||]
- 12:43 PM
- Smiley face icon
- Bell icon
- [12:43:05]

index.js — ttd-demo

EXPLORER

OPEN EDITORS

- launch.json .vscode
- index.js

TTD-DEMO

- _diagnosticTraces
- _ttd_log_
- .node-chakracore
- .vscode
- launch.json
- index.js

lauch.json index.js

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

PROBLEMS OUTPUT TERMINAL ...

1: zsh

Recording started (after main module loaded)...
Write error trace to: /Users/mmarchini/workspace/tmp/ttd-demo/_diagnosticTraces/emitOnException pid48619
/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9
crash();
^

ReferenceError: 'crash' is not defined
at bar (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9:5)
at Anonymous function (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:12:20)
at ontimeout (timers.js:426:5)
at tryOnTimeout (timers.js:289:5)
at listOnTimeout (timers.js:252:5)
at processTimers (timers.js:212:3)

λ [workspace/tmp/ttd-demo]
→

[12:43:05]

Ln 13, Col 1 Spaces: 2 UTF-8 LF JavaScript 49% [|||||] 12:43 PM ☺ 🔔

0 0 DOCKER BOOKMARKS

index.js — ttd-demo

DEBUG Time-T

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions

launch.json index.js

```
1 function foo(x) {  
2     return x == null;  
3 }  
4  
5 function bar(x) {  
6     const shouldCrash = foo(x);  
7     if (shouldCrash)  
8         crash();  
9 }  
10  
11 setTimeout(() => { bar(null); });  
12  
13
```

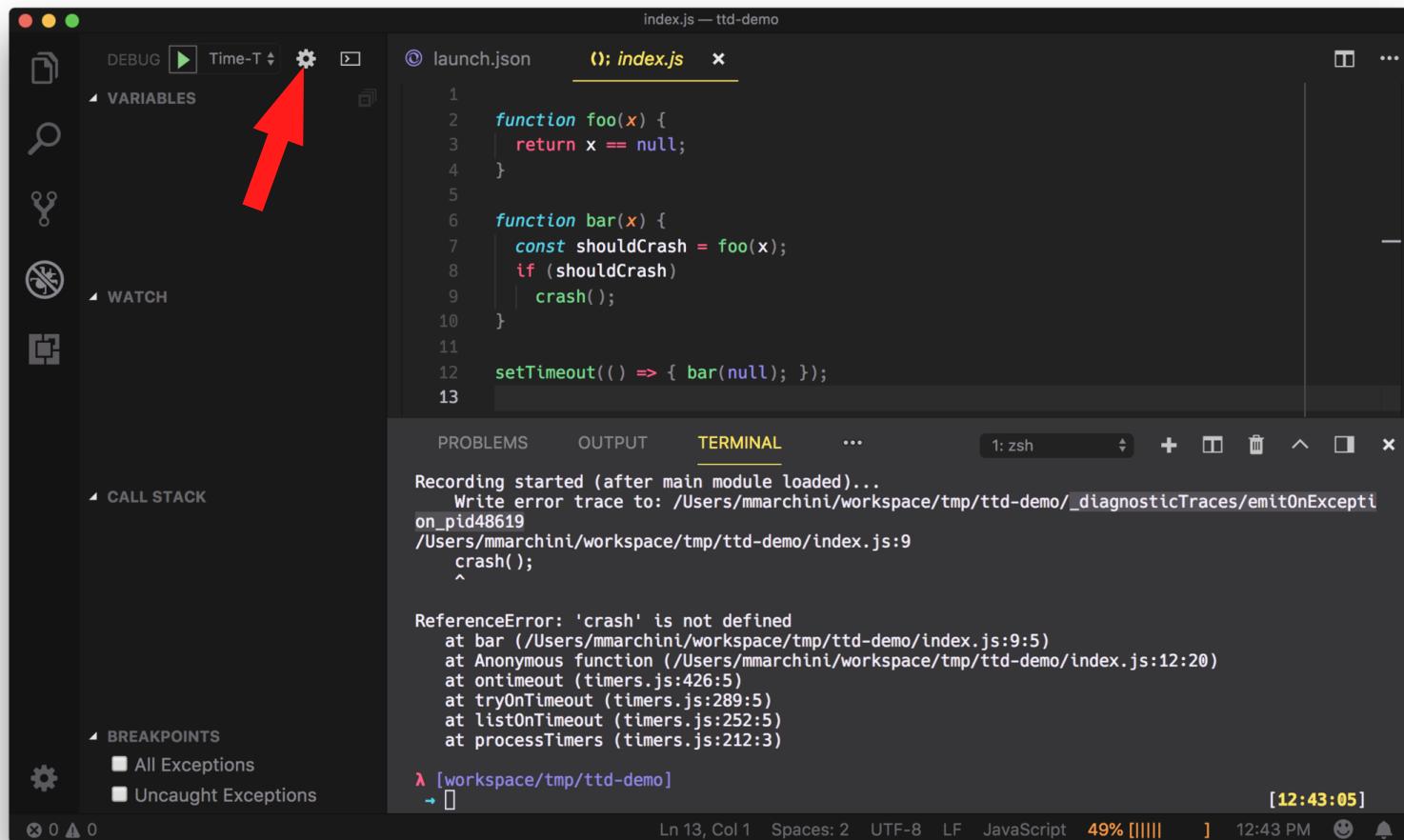
PROBLEMS OUTPUT TERMINAL ...

1: zsh

```
Recording started (after main module loaded)...  
Write error trace to: /Users/mmarchini/workspace/tmp/ttd-demo/_diagnosticTraces/emitOnException_pid48619  
/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9  
    crash();  
    ^  
  
ReferenceError: 'crash' is not defined  
    at bar (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9:5)  
    at Anonymous function (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:12:20)  
    at ontimeout (timers.js:426:5)  
    at tryOnTimeout (timers.js:289:5)  
    at listOnTimeout (timers.js:252:5)  
    at processTimers (timers.js:212:3)  
  
λ [workspace/tmp/ttd-demo]  
→ []
```

[12:43:05]

Ln 13, Col 1 Spaces: 2 UTF-8 LF JavaScript 49% [|||||] 12:43 PM



index.js — ttd-demo

DEBUG ⏪ Time-T ⚙️

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions

launch.json

index.js

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

PROBLEMS OUTPUT TERMINAL

1: zsh

```
Recording started (after main module loaded)...
Write error trace to: /Users/mmarchini/workspace/tmp/ttd-demo/_diagnosticTraces/emitOnException_pid48619
/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9
    crash();
    ^
ReferenceError: 'crash' is not defined
  at bar (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9:5)
  at Anonymous function (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:12:20)
  at ontimeout (timers.js:426:5)
  at tryOnTimeout (timers.js:289:5)
  at listOnTimeout (timers.js:252:5)
  at processTimers (timers.js:212:3)
```

λ [workspace/tmp/ttd-demo]

[12:43:05]

Ln 13, Col 1 Spaces: 2 UTF-8 LF JavaScript 49% [|||||] 12:43 PM

launch.json — tdd-demo

DEBUG Trace ⚡ ⚙️ ⌛

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions

launch.json x index.js

```
8     "name": "Trace Debug",
9     "type": "node",
10    "request": "launch",
11    "runtimeExecutable": "${env:HOME}/.nvs/nvs",
12    "runtimeArgs": [
13      "run",
14      "chakracore-nightly"
15      "--noLazy",
16      "--break-first",
17      "--replay-debug=${workspaceRoot}/_diagnosticTraces/emitOnException_pid48619"
18    ],
19    "console": "internalConsole"
20 }
```

Add Configuration...

PROBLEMS 1 OUTPUT TERMINAL ...

Recording started (after main module loaded)...

Write error trace to: /Users/mmarchini/workspace/tmp/tdd-demo/_diagnosticTraces/emitOnException_pid48619

/Users/mmarchini/workspace/tmp/tdd-demo/index.js:9
crash();
^

ReferenceError: 'crash' is not defined

at bar (/Users/mmarchini/workspace/tmp/tdd-demo/index.js:9:5)
at Anonymous function (/Users/mmarchini/workspace/tmp/tdd-demo/index.js:12:20)
at ontimeout (timers.js:426:5)
at tryOnTimeout (timers.js:289:5)
at listOnTimeout (timers.js:252:5)
at processTimers (timers.js:212:3)

λ [workspace/tmp/tdd-demo]
→ []

[12:43:05]

Ln 17, Col 42 (42 selected) Spaces: 2 UTF-8 LF JSON with Comments 49% [|||||] 12:43 PM

The screenshot shows the VS Code interface with a dark theme. On the left, the sidebar contains sections for VARIABLES, WATCH, CALL STACK, and BREAKPOINTS, each with expandable arrows. The main area displays a file named `launch.json` with the following content:

```
8      "name": "Trace Debug",
9      "type": "node",
10     "request": "launch",
11     "runtimeExecutable": "${env:HOME}/.nvs/nvs",
12     "runtimeArgs": [
13       "run",
14       "chakracore-nightly"
15       "--noLazy",
16       "--break-first",
17       "--replay-debug=${workspaceRoot}/_diagnosticTraces/emitOnException_pid48619"
18     ],
19     "console": "internalConsole"
20 }
```

A red box highlights the line `--replay-debug=${workspaceRoot}/_diagnosticTraces/emitOnException_pid48619`. A tooltip for "Add Configuration..." is visible near the bottom right of the configuration area.

Below the editor, the terminal tab is active, showing the output of a recording session:

```
Recording started (after main module loaded)...
Write error trace to: /Users/mmarchini/workspace/tmp/ttd-demo/_diagnosticTraces/emitOnException_pid48619
/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9
    crash();
^

ReferenceError: 'crash' is not defined
at bar (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:9:5)
at Anonymous function (/Users/mmarchini/workspace/tmp/ttd-demo/index.js:12:20)
at ontimeout (timers.js:426:5)
at tryOnTimeout (timers.js:289:5)
at listOnTimeout (timers.js:252:5)
at processTimers (timers.js:212:3)
```

The status bar at the bottom shows the current file is `index.js`, line 17, column 42, with 42 selected characters. It also shows the file is in JSON format, has 49% completion, and was last updated at 12:43 PM. The status bar ends with a smiley face icon and a bell icon.

index.js — tdd-demo

DEBUG ▶ Trace ↴ ⚙️ ⌂ ...

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js 8

✖ 1 ⚡ 0 ▶ Trace Debug (tdd-demo) Auto Attach: Off Ln 7, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:45 PM 😊 📲

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

A screenshot of the Visual Studio Code interface, specifically the Debug viewlet, which is part of the Trace Debug (tdd-demo) session. The code editor shows a file named `index.js` with the following content:

```
function foo(x) {
  return x == null;
}

function bar(x) {
  const shouldCrash = foo(x);
  if (shouldCrash)
    crash();
}

setTimeout(() => { bar(null); });


```

The line `if (shouldCrash)` is highlighted with a red arrow pointing to it from the left side of the screen. This indicates that the debugger has stopped at this specific line of code. The status bar at the bottom of the interface shows the current line number (Ln 7, Col 3), character count (Spaces: 2), encoding (UTF-8 LF), language (JavaScript), and zoom level (48% [|||||]).

index.js — ttd-demo

DEBUG ▶ Trace ↴ ⚙️ ⌂ ...

VARIABLES

WATCH

CALL STACK

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js 8

✖ 1 ⚡ 0 ▶ Trace Debug (ttd-demo) Auto Attach: Off Ln 7, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:45 PM 😊 📲

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

index.js — tdd-demo

DEBUG Trace ▾ ⚙️ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - shouldCrash: true
 - x: null
- WATCH

CALL STACK PAUSED ON BREAK...

- bar index.js 8:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js 8

✖ 1 ⚠ 0 Trace Debug (tdd-demo) Auto Attach: Off Ln 8, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:45 PM 😊 📲

```
1  function foo(x) {
2    return x == null;
3  }
4
5
6  function bar(x) {
7    const shouldCrash = foo(x);
8    if (shouldCrash)
9      crash();
10 }
11
12 setTimeout(() => { bar(null); });
13
```

index.js — ttd-demo

DEBUG ▶ Trace ⌂ ⚙ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - shouldCrash: true
 - x: null
- WATCH

CALL STACK PAUSED ON BREAKPOINT

- bar index.js 8:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js 8

✖ 1 ⚠ 0 Trace Debug (ttd-demo) Auto Attach: Off Ln 8, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:45 PM ☺ 🔔

```
1  function foo(x) {
2    return x == null;
3  }
4
5
6  function bar(x) {
7    const shouldCrash = foo(x);
8    if (shouldCrash)
9      crash();
10 }
11
12 setTimeout(() => { bar(null); });
13
```

A screenshot of the Visual Studio Code interface during a debugging session. The title bar shows "index.js — tdd-demo". The left sidebar contains the "VARIABLES" and "WATCH" sections, which are currently empty. The main editor area displays the following code:

```
function foo(x) {
    return x == null;
}

function bar(x) {
    const shouldCrash = foo(x);
    if (shouldCrash)
        crash();
}

setTimeout(() => { bar(null); }, 0);
```

The line 8, where the condition "if (shouldCrash)" is evaluated, is highlighted with a yellow background. A red arrow points to this line from the left sidebar. The bottom status bar shows "Ln 8, Col 3" and "JavaScript".

The screenshot shows the VS Code interface in debug mode. The title bar indicates the file is `index.js` and the workspace is `tdd-demo`. The top navigation bar includes icons for DEBUG, Trace, settings, and a stop button.

The left sidebar contains the `VARIABLES` panel, which shows local variables `this`, `arguments`, `shouldCrash` (set to `true`), and `x` (set to `null`). It also includes sections for `WATCH` and `CALL STACK`.

The main editor area displays the code:

```
1 function foo(x) {
2     return x === null;
3 }
4
5 function bar(x) {
6     const shouldCrash = foo(x);
7     if (shouldCrash)
8         crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

A red arrow points to the green play button icon in the toolbar, indicating the current state is paused. Another red arrow points to the yellow circular breakpoint icon at line 8, column 3, where the code `crash();` is located.

The bottom status bar shows the current line is `Ln 8, Col 3`, with `Spaces: 2`, `UTF-8`, `LF`, `JavaScript`, and a zoom level of `48% [|||||]`. The timestamp is `12:45 PM`, and there are icons for smiley face, notifications, and a bell.

index.js — tdd-demo

DEBUG ▶ Trace ⌂ ⚙ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - shouldCrash: true
 - x: null
- WATCH

CALL STACK PAUSED ON BREAKPOINT

- bar index.js 9:5
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js

8

✖ 1 ⚠ 0 Trace Debug (tdd-demo) Auto Attach: Off Ln 9, Col 5 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:46 PM ☺ 🔔

```
1  function foo(x) {
2    | return x == null;
3  }
4
5
6  function bar(x) {
7    const shouldCrash = foo(x);
8    if (shouldCrash)
9      crash();
10 }
11
12 setTimeout(() => { bar(null); });
13
```

A screenshot of the Visual Studio Code (VS Code) interface, specifically the Debug view, showing a JavaScript file named `index.js`. The code contains a function `bar` which calls `foo` with a null argument and then calls `crash`. A red arrow points to the step back button (`<`) in the top toolbar.

```
index.js — tdd-demo
1  function foo(x) {
2      return x == null;
3  }
4
5
6  function bar(x) {
7      const shouldCrash = foo(x);
8      if (shouldCrash)
9          crash();
10 }
11
12 setTimeout(() => { bar(null); });
13
```

The left sidebar shows the `VARIABLES` panel with local variables `this`, `arguments`, `shouldCrash`, and `x`. The `WATCH` panel is also visible. The `CALL STACK` panel shows the current stack trace, which is paused on a breakpoint in `index.js`. The `BREAKPOINTS` panel lists breakpoints for `All Exceptions`, `Uncaught Exceptions`, and `index.js`.

Bottom status bar: `Ln 9, Col 5`, `Spaces: 2`, `UTF-8`, `LF`, `JavaScript`, `48% [|||||]`, `12:46 PM`, `Trace Debug (tdd-demo)`, `Auto Attach: Off`.

index.js — tdd-demo

DEBUG Trace ▾ ⚙️ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - shouldCrash: true
 - x: null
- WATCH

CALL STACK PAUSED ON BREAK...

- bar index.js 8:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js

8

index.js

```
1 function foo(x) {
2   return x == null;
3 }
4
5 function bar(x) {
6   const shouldCrash = foo(x);
7   if (shouldCrash)
8     crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

Ln 8, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:46 PM ☺ 🔔

Trace Debug (tdd-demo) Auto Attach: Off

index.js — tdd-demo

DEBUG ▶ Trace ⌂ ⚙ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - shouldCrash: true
 - x: null
- WATCH

launch.json

```
1 function foo(x) {  
2     return x == null;  
3 }  
4  
5 function bar(x) {  
6     const shouldCrash = foo(x);  
7     if (shouldCrash)  
8         crash();  
9     }  
10 }  
11  
12 setTimeout(() => { bar(null); });  
13 }
```

CALL STACK PAUSED ON BREAKPOINT

- bar index.js 8:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js

3

Trace Debug (tdd-demo) Auto Attach: Off Ln 8, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:46 PM ☺ 🔔

A screenshot of the Visual Studio Code (VS Code) interface, specifically the Debug view. The code editor shows a file named `index.js` with the following content:

```
1 function foo(x) {
2     return x == null;
3 }
4
5 function bar(x) {
6     const shouldCrash = foo(x);
7     if (shouldCrash)
8         crash();
9 }
10
11 setTimeout(() => { bar(null); });
12
13
```

The line `if (shouldCrash)` is highlighted with a yellow background, indicating it is the current line of execution. A red arrow points from the left margin towards the `VARIABLES` panel.

The `VARIABLES` panel on the left displays the following local variables:

- `this: {...}`
- `arguments: {...}`
- `shouldCrash: true`
- `x: null`

The `CALL STACK` panel shows the call stack for the paused execution:

- bar index.js 8:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js
- listOnTimeout timers.js

The `BREAKPOINTS` panel shows a list of breakpoints:

- All Exceptions
- Uncaught Exceptions
- index.js

At the bottom, the status bar shows: `Ln 8, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:46 PM`.

A screenshot of the Visual Studio Code (VS Code) interface, specifically the Debug view, showing a paused JavaScript execution. The code being debugged is located in `index.js`.

The code in `index.js` is:

```
function foo(x) {
  return x == null;
}

function bar(x) {
  const shouldCrash = foo(x);
  if (shouldCrash)
    crash();
}

setTimeout(() => { bar(null); }, 0);
```

The debugger is currently at line 8, column 3, where the condition `if (shouldCrash)` is being evaluated.

The interface includes the following sections:

- VARIABLES**: Shows local variables `this`, `arguments`, `shouldCrash`, and `x`. `x` is set to `null`. A red arrow points to this section.
- CALL STACK**: Shows the call stack: `bar` at index.js:8:3, followed by several entries from the Node.js timers module.
- BREAKPOINTS**: Shows breakpoints for `All Exceptions`, `Uncaught Exceptions`, and the current file `index.js` at line 8. A red arrow points to this section.

At the bottom, the status bar displays: `Ln 8, Col 3`, `Spaces: 2`, `UTF-8`, `LF`, `JavaScript`, `48% [|||||]`, `12:46 PM`, and icons for smiley face, bell, and other notifications.

index.js — ttd-demo

DEBUG ▶ Trace ⌂ ⚙ ⚡

VARIABLES

- Local
 - this: {...}
 - arguments: {...}
 - x: null
- Closure
- WATCH

launch.json

```
1 function foo(x) {  
2     return x == null;  
3 }  
4  
5 function bar(x) {  
6     const shouldCrash = foo(x);  
7     if (shouldCrash)  
8         crash();  
9     }  
10 }  
11  
12 setTimeout(() => { bar(null); });  
13 }
```

CALL STACK PAUSED ON BREAKPOINT

- foo index.js 3:3
- bar index.js 7:3
- Anonymous function index.js
- ontimeout timers.js 426:5
- tryOnTimeout timers.js

BREAKPOINTS

- All Exceptions
- Uncaught Exceptions
- index.js

Ln 3, Col 3 Spaces: 2 UTF-8 LF JavaScript 48% [|||||] 12:46 PM

Trace Debug (ttd-demo) Auto Attach: Off

Extension: NodeChakra Time Travel Debug — ttd-demo

EXTENSIONS ...

INSTALLED 17

- Beautiful functionality for prof...
monokai
- NodeChakra Time Trav...** 0.1.3
Node debugger with time-trav...
ttd-trace-tools
- Python 2018 3.1
RECOMMENDED 7
- Debugger for Chrome 4.3.0
Debug your JavaScript code i...
Microsoft
- Beautify 1.3.0
Beautify code in place for VS ...
HoakyQR
- jshint 0.10.18
Integrates JSHint into VS Cod...
Dirk Baeumer
- ESLint 1.4.8
Integrates ESLint into VS Cod...
Dirk Baeumer
- C++ Intellisense 0.2.2
C/C++ Intellisense with the he...
austin

Search Extensions in Marketplace

launch.json

Extension: NodeChakra Time Travel Debug

NodeChakra Time Travel Debug ttd-trace-tools.node...

ttd-trace-tools | ⚡ 137 | ★★★★★ | Repository | License

Node debugger with time-travel support

Disable Uninstall

[Details](#) [Contributions](#) [Changelog](#) [Dependencies](#)

Node Debugger with mixed Live and Time-Travel support

This debugger provides a launch configurations and support for mixing live and time-travel debugging in Node.js. In addition to the Visual Studio Code debugger logic this extension provides:

1. **NodeChakraCore binaries** with time-travel debugging functionality.
2. Launch configuration for mixed live/time-travel debugging.

Notes

1. The debugger always uses the extension provided NodeChakraCore binaries. If your applicatio

depends on a specific version of Node you may encounter unusual behavior.

2. Time-Travel mode is **not** enabled until synchronous module loading has completed and Node is

45% [|||||] 12:50 PM

×

1 0 ► Trace Debug (ttd-demo) Auto Attach: Off

The screenshot shows the Visual Studio Code interface with the following details:

- Extensions Sidebar:** On the left, the Extensions sidebar is open, displaying a list of installed and recommended extensions. The "INSTALLED" section includes "monokai", "NodeChakra Time Trav...", "ttd-trace-tools", and "Python". The "RECOMMENDED" section includes "Debugger for Chrome", "Beautify", "jshint", "ESLint", and "C++ Intellisense".
- File Explorer:** The main workspace area shows a file named "launch.json" under the "tdd-demo" folder.
- Code Editor:** The code editor displays the "launch.json" file content, which defines two configurations for debugging Node.js applications using the NodeChakra Time Travel Debugger:

```
1  {
2    // Use IntelliSense to learn about possible attributes.
3    // Hover to view descriptions of existing attributes.
4    // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5    "version": "0.2.0",
6    "configurations": [
7      {
8        "name": "Time-Travel Live",
9        "type": "node-chakracore-time-travel-debugger",
10       "request": "launch",
11       "program": "${workspaceFolder}/index.js",
12       "cwd": "${workspaceFolder}"
13     },
14     {
15       "name": "Trace Debug",
16       "type": "node",
17       "request": "launch",
18       "runtimeExecutable": "${env:HOME}/.nvs/nvs",
19       "runtimeArgs": [
20         "run",
21         "chakracore-nightly"
22         "--noLazy",
23         "--break-first",
24         "--replay-debug=${workspaceRoot}/_diagnosticTraces/emitOnException_pid48619"
25       ],
26       "console": "internalConsole"
27     }
28   ]
29 }
```
- Status Bar:** The bottom status bar shows the following information: Line 14, Column 1 (210 selected), Spaces: 2, UTF-8, LF, JSON with Comments, 45% [|||||], 12:51 PM, a smiley face icon, and a bell icon.

Time-Travel Debugging

Current Status

Time-Travel Debugging

Current Status

- Apenas com node-chakracore + VS Code

Time-Travel Debugging

Current Status

- Apenas com node-chakracore + VS Code
 - Protocolo utilizado é aberto

Time-Travel Debugging

Current Status

- Apenas com node-chakracore + VS Code
 - Protocolo utilizado é aberto
 - Colaboração com V8 para trazer essa funcionalidade para o Node.js Core

ChakraCore vai substituir o V8?



ChakraCore vai substituir o V8?

Não é o objetivo

**Existe "briga" entre
as VMs?**

Existe "briga" entre as VMs?

**Existe uma competitividade
saudável**

Existe "briga" entre as VMs?

Existe uma competitividade
saudável

Também existe muita
colaboração entre elas

Posso usar node-chakracore em produção?

Posso usar node-chakracore em produção?

Não deve ter problemas

Posso usar node-chakracore em produção?

Não deve ter problemas

Use a seu próprio risco

Posso usar TTD em produção?

Posso usar TTD em produção?

Overhead de memória ~10%

Overhead de processamento

Posso usar TTD em produção?

Overhead de memória ~10%

Overhead de processamento

Não recomendado no momento, mas é um objetivo futuro



www.sthima.com



@mmarkini



<http://mmarchini.me/>



oss@mmarchini.me



<https://github.com/mmarchini/>