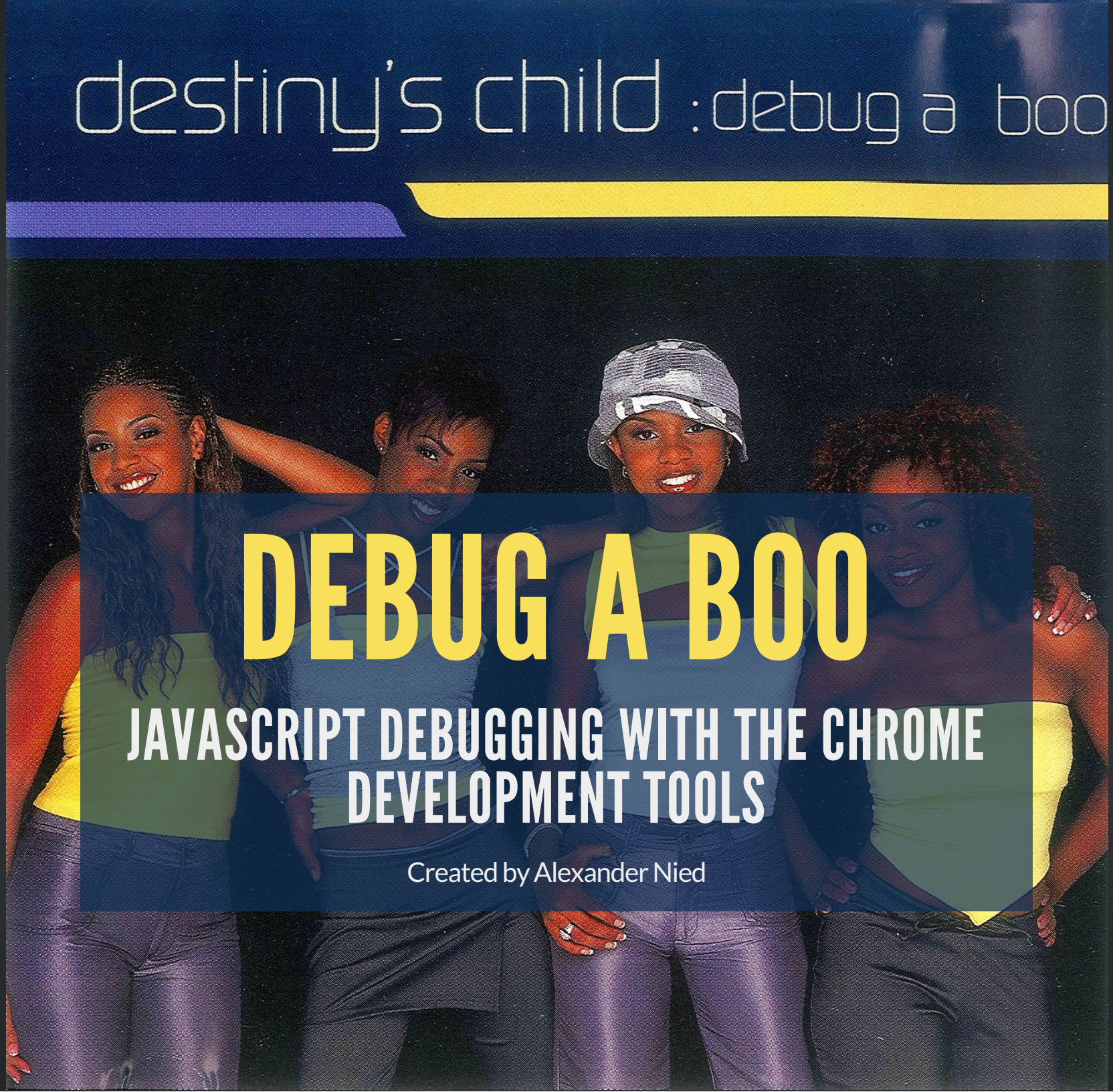


DEBUG A BOO

JAVASCRIPT DEBUGGING WITH THE CHROME DEVELOPMENT TOOLS

Created by Alexander Nied

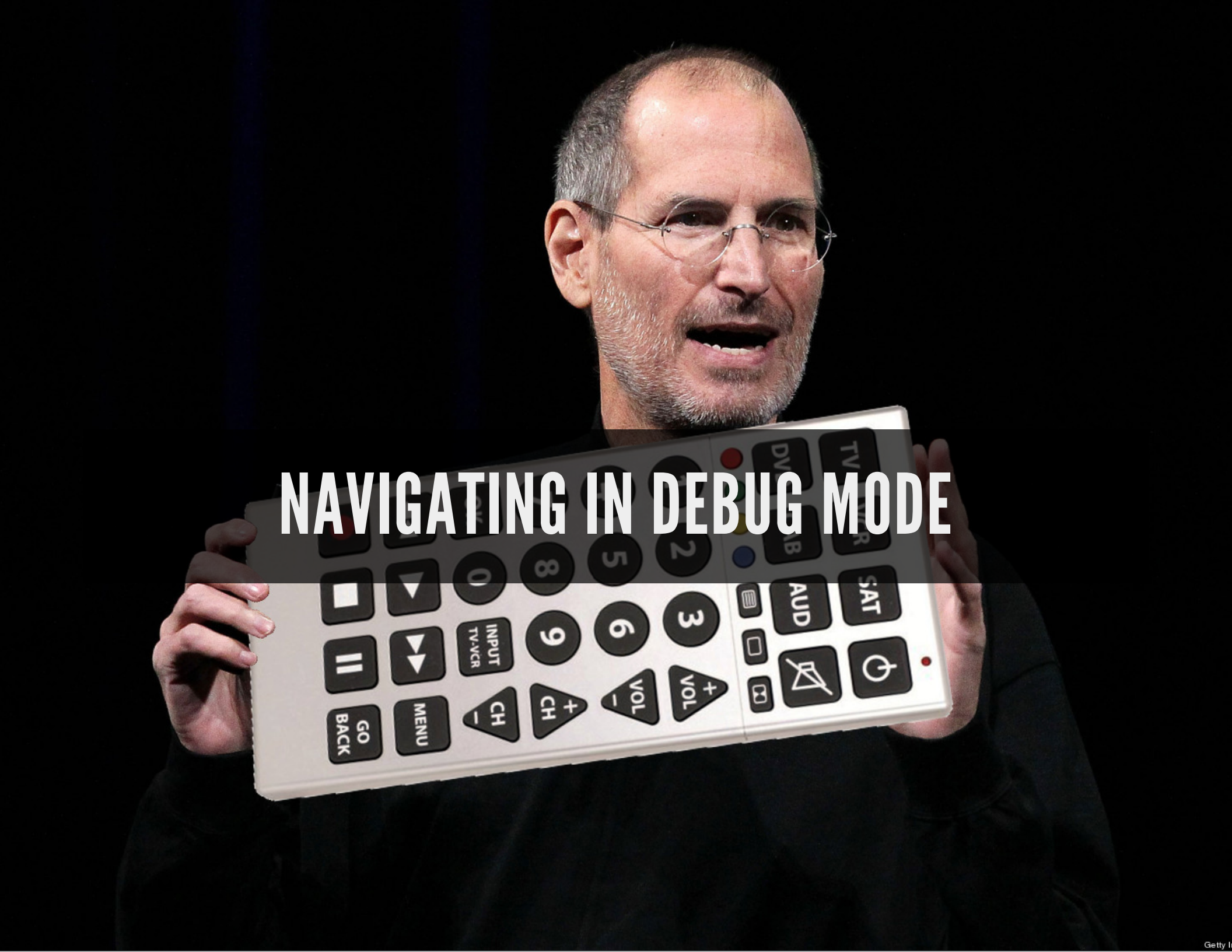

DEBUGGING JAVASCRIPT USED TO BE HARD.

BUT NOW DEBUGGING JAVASCRIPT IS ~~EASY!~~ EASIER

BASIC BREAKPOINTS

BASIC BREAKPOINT MECHANICS

- Add/Remove
- Enable/Disable
- Group Enable/Disable


NAVIGATING IN DEBUG MODE
INPUT
TV-VCR
MENU
GO BACK
CH
CH
VOL
VOL
AUD
SAT

PAUSE

Just what it sounds like-- pauses execution at exactly the moment you push it. As you can imagine, this has pretty limited usefulness. Much better to enter debug mode using breakpoints or some other more controlled method.

CONTINUE/PLAY

When you *are* in a paused state, pushing the `continue` button allows scripts to continue execution as normal, until they hit the next breakpoint.

Additionally, the Chrome dev tools have an additional `play` button which will continue and ignore all breakpoints for 500ms-- perfect for allowing a page to continue loading when you have a breakpoint in a `for` loop. The `play` button can be accessed by clicking and holding on the `continue` button.

STEP OVER

Somewhat confusing name, the `step over` button executes the current command and moves on to the next one *in the current scope*. If you have reached the end of code execution in the current scope, `step over` will then move up the call stack to the origination point of the current execution point.

STEP IN

`step in` will move down the call stack when there is an oppotunity to do so and begin debugging the function being called from that line-- if there is no new call being executed from that line, it will behave like a simple `step over`.

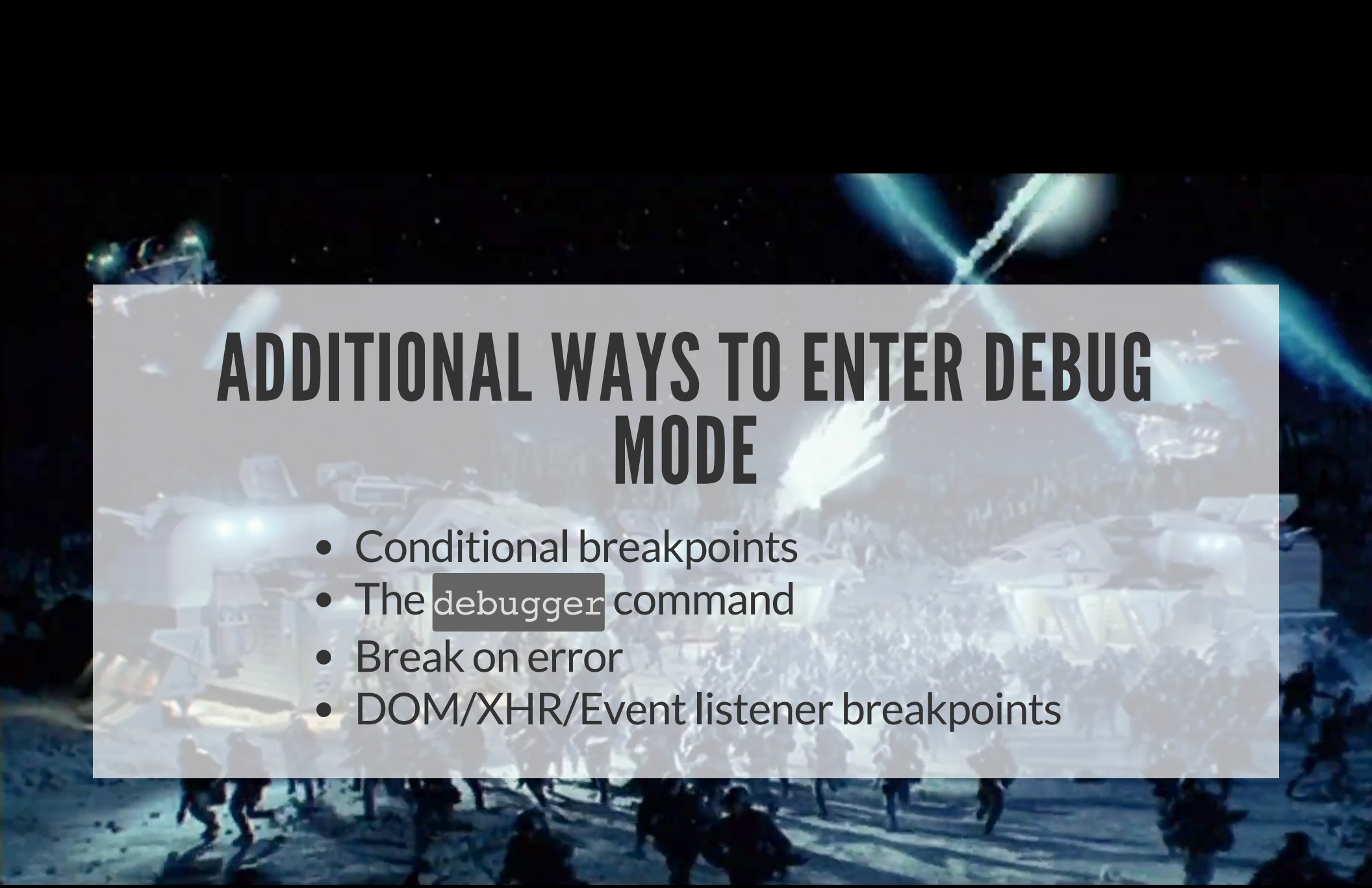
STEP OUT

`step out` is the opposite of `step in`-- rather than continuing down commands in the current function on the call stack, it instead backs out out of the current scope to the point of the originating call to the current scope.

FIGHTING THE BUGS

"The only good bug is a dead bug."


THE TOOLS AT YOUR DISPOSAL
• Hover
• Watch list
• Console
• Change and save
WOULD YOU LIKE TO KNOW MORE?



ADDITIONAL WAYS TO ENTER DEBUG MODE

- Conditional breakpoints
- The `debugger` command
- Break on error
- DOM/XHR/Event listener breakpoints

NETWORK STACK TRACING

You can also find out how any given downloaded resource was initiated for download by going to the Network Panel--here you can hover over the blue underlined text in the initiator colum to see a navigable stack trace of showing the steps that led up to the resource being requested. When using libraries like Backbone and jQuery there will likely be many library-internal function calls for things like ajax/fetch requests, but sorting out your personal code from library code shouldn't prove too difficult.

THE CONSOLE: MORE THAN JUST LOGGING

The console object has many methods beyond the oft used `log` method-- including tracing, grouping, errors, and more. This API is not standardized and may differ from browser to browser-- for chrome it can be found at:

https://developer.chrome.com/devtools/docs/console-api

LOCAL FILES

You can bring your local files directly into the Chrome debugger.

That's all I got.

What do you want, man, it's like 2:30AM...


JAVASCRIPT DEBUGGING:
IT'S NOT JUST YOUR JOB...
...IT'S YOUR DESTINY.
(This is another picture of Destiny's Child, if you didn't know...)

fin