DO-EXPRESSIONS

Status update

Dave Herman & Chris Krycho

REFRESHER

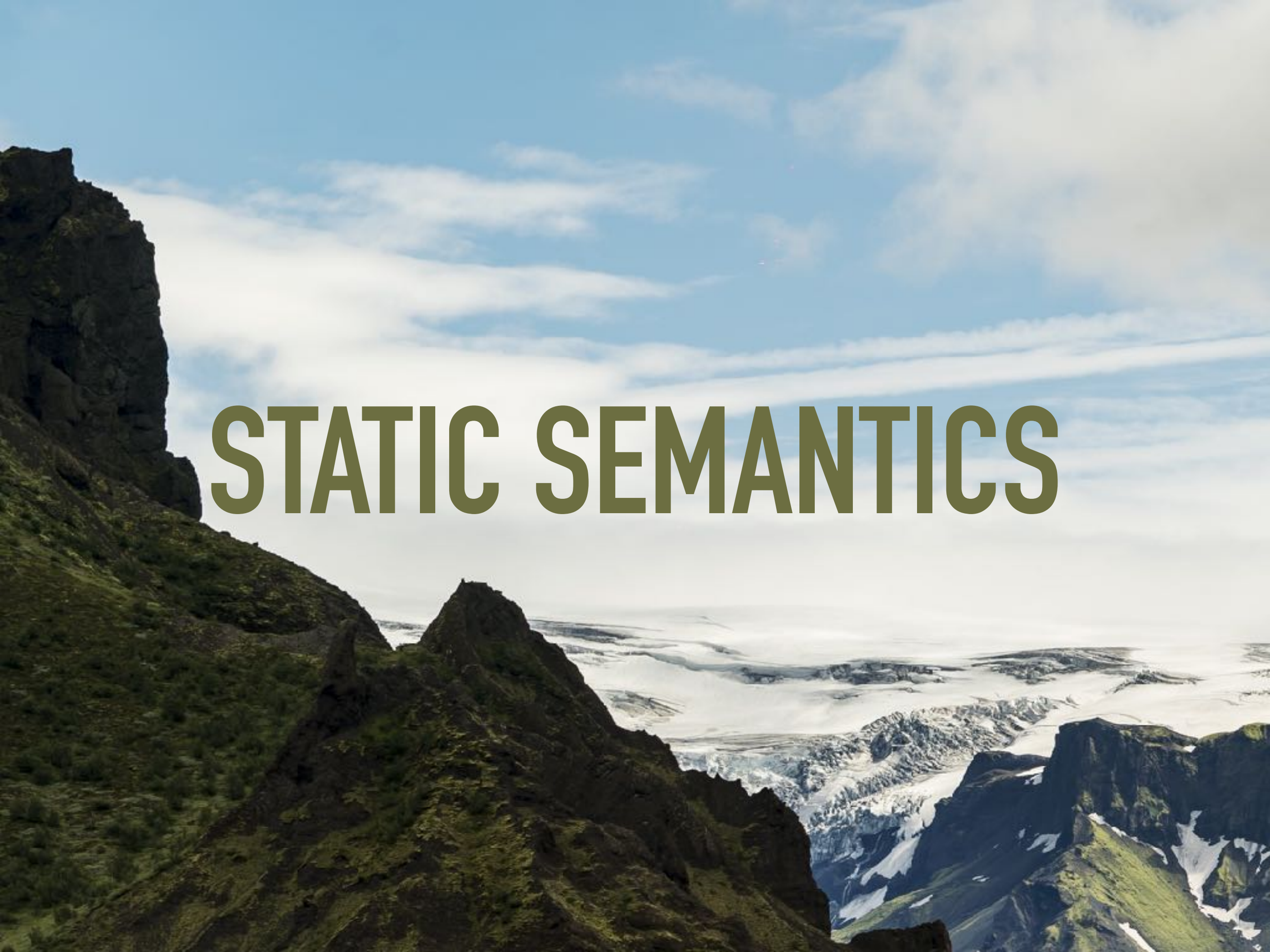
```
let file = do {
  let parent = path.parent();

  if (parent) {
    await mkdirp(parent);
  }
  await createFile(dest.join(path))
};
```

“Tennent’s Correspondence Principle” (TCP)

```
async function countDependencies {
  let search = do {
    if (!await fs.exists('node_modules'))
      return 0;
    await subdirs(‘node_modules')
  };
  while (search.length > 0) {
    // ...
  }
}
```

STATUS UPDATE

- ~~Syntax~~ more another time!
- Static semantics
- Dynamic semantics



STATIC SEMANTICS

Observation: do expressions allow statements to appear in new syntactic positions.

Question: are there positions where

```
do { $stmt }
```

can introduce new interactions with the surrounding context?

"INTERESTING" STATEMENTS AND POSITIONS

- **`return`**, **`var`**
  - Parameter default expressions
- **`break`**, **`continue`**
  - Parameter default expressions
  - Loop headers

RETURN IN DEFAULT EXPRESSIONS

```
function f(x = do { return 42; }) {
    return x;
}
```

Options:

1. Return from outer function
2. Return from inner function
3. Disallow

RETURN IN DEFAULT EXPRESSIONS: OUTER?

```
function outer() {
    function f(x = do { return 42; }) {
        return x;
    }
    return f();
}
outer(); // error: outer already returned!
```

implicitly delayed

RETURN IN DEFAULT EXPRESSIONS: OUTER?

```
function outer() {

    inner = {|| return 42; };

}
outer();
inner(); // error: outer already returned!
```

hypothetical TCP function

IOW, a design goal of do: TCP without “double returns”

RETURN IN DEFAULT EXPRESSIONS: INNER?

```
function f(x = do { return 42; }) {
    return x;
}
let a = f(17); // 17
let b = f();   // 42
```

RETURN IN DEFAULT EXPRESSIONS: INNER?

```
var y = 'outer';

function f(x = () => y) {

    var y = 'inner';

    return x();

}


f(); // 'outer'
```

separate scope from body

is **var** analogous to **return**?

RETURN IN DEFAULT EXPRESSIONS: DISALLOW?

```
function f(x = do { return 42; }) {
    return x;
}

// syntax error: return not in function
```

RETURN IN DEFAULT EXPRESSIONS: RECAP

```
function f(x = do { return 42; }) {
    return x;
}
```

Options:

1. ~~Return from outer function~~ ✗
2. Return from inner function
3. Disallow

VAR IN DEFAULT EXPRESSIONS

```
function f(x = do { var y = 17; 42 }) {
    return x + y;
}
```

Options:

1. ~~Bind in outer function~~ ✗
2. Bind in inner function
3. Disallow

BREAK/CONTINUE IN DEFAULT EXPRESSIONS: BAD!

```
function outer() {
  while (...) {
    inner = function f(do { break; }) {
      ...
    };
  }
}
outer();
inner(); // loop already terminated!
```

BREAK/CONTINUE TO LABEL IN LOOP HEADER: GOOD!

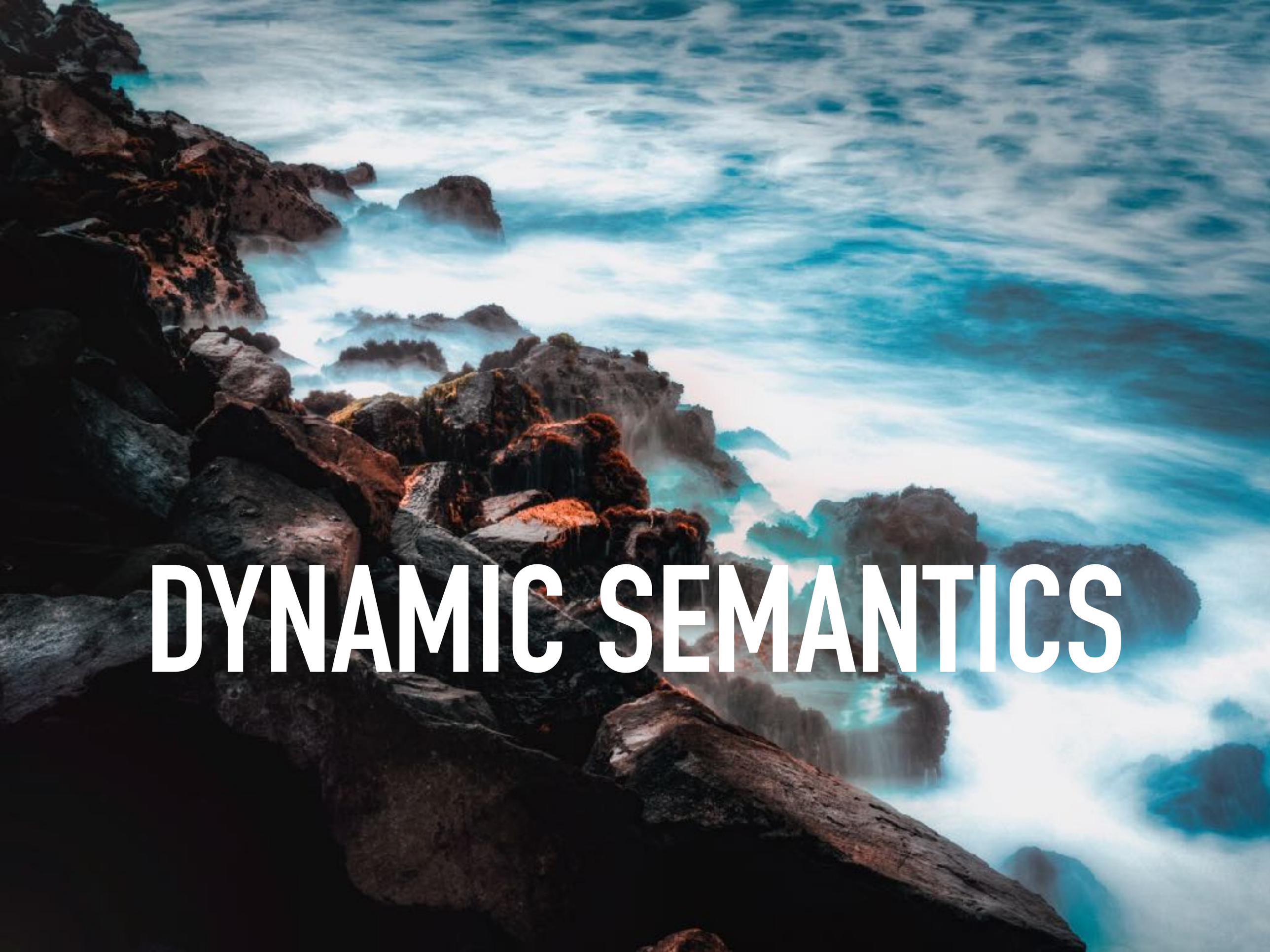
```
outer:
  while (...) {
    while (do { break outer; }) {
      ...
    }

    while (do { continue outer; }) {
      ...
    }
  }
```

BREAK/CONTINUE WITHOUT LABEL IN LOOP HEADER? HM...

```
outer:
  while (...) {
    while (do { break; }) {
      ...
    }
  }
```

Options:

1. Break from outer loop
2. Break from inner loop
3. Disallow



DYNAMIC SEMANTICS

ALTERNATIVE COMPLETIONS?

```
let a = [ ... ];
let x = do {
  for (let i = 0; i < a.length; i++) {
    a[i] * 2
  }
};
```

Options:

1. Use existing language's completion values
2. New concept of do-completions?

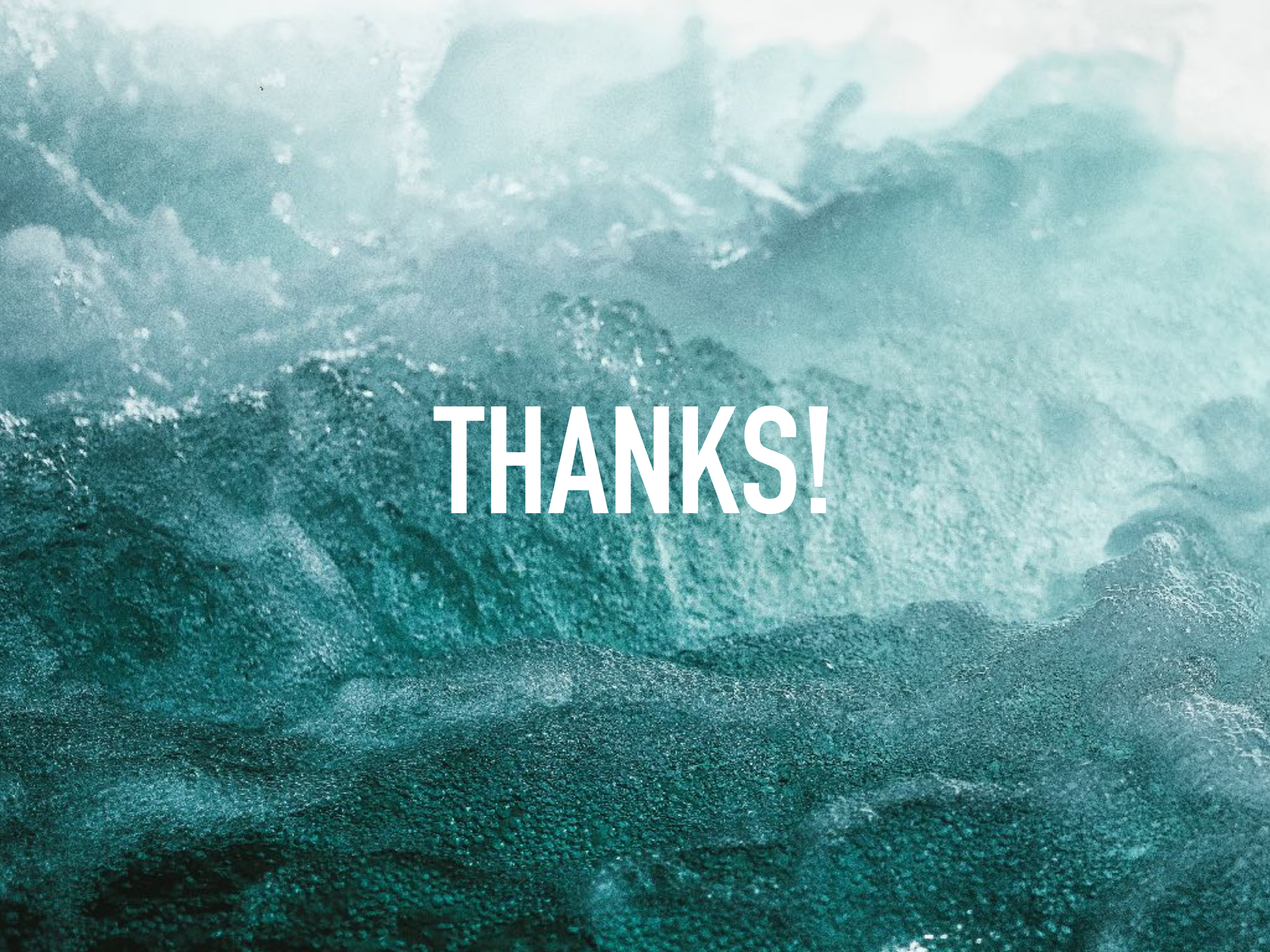

THANKS!

IMAGE CREDITS

Title slide photo by Antonio Rossi on Unsplash

https://unsplash.com/photos/naUC8KzyXoo

Refresher slide photo by Pawel Czerwiński on Unsplash

https://unsplash.com/photos/Zf5YlpPw_kE

Static semantics slide photo by Jonatan Pie on Unsplash

https://unsplash.com/photos/A1LEJwIeabU

Dynamic semantics slide photo by Giancarlo Revolledo on Unsplash

https://unsplash.com/photos/Ef32LHjkw64

Final slide photo by eberhard grossesteiger on Unsplash

https://unsplash.com/photos/EcVGogpC1G4