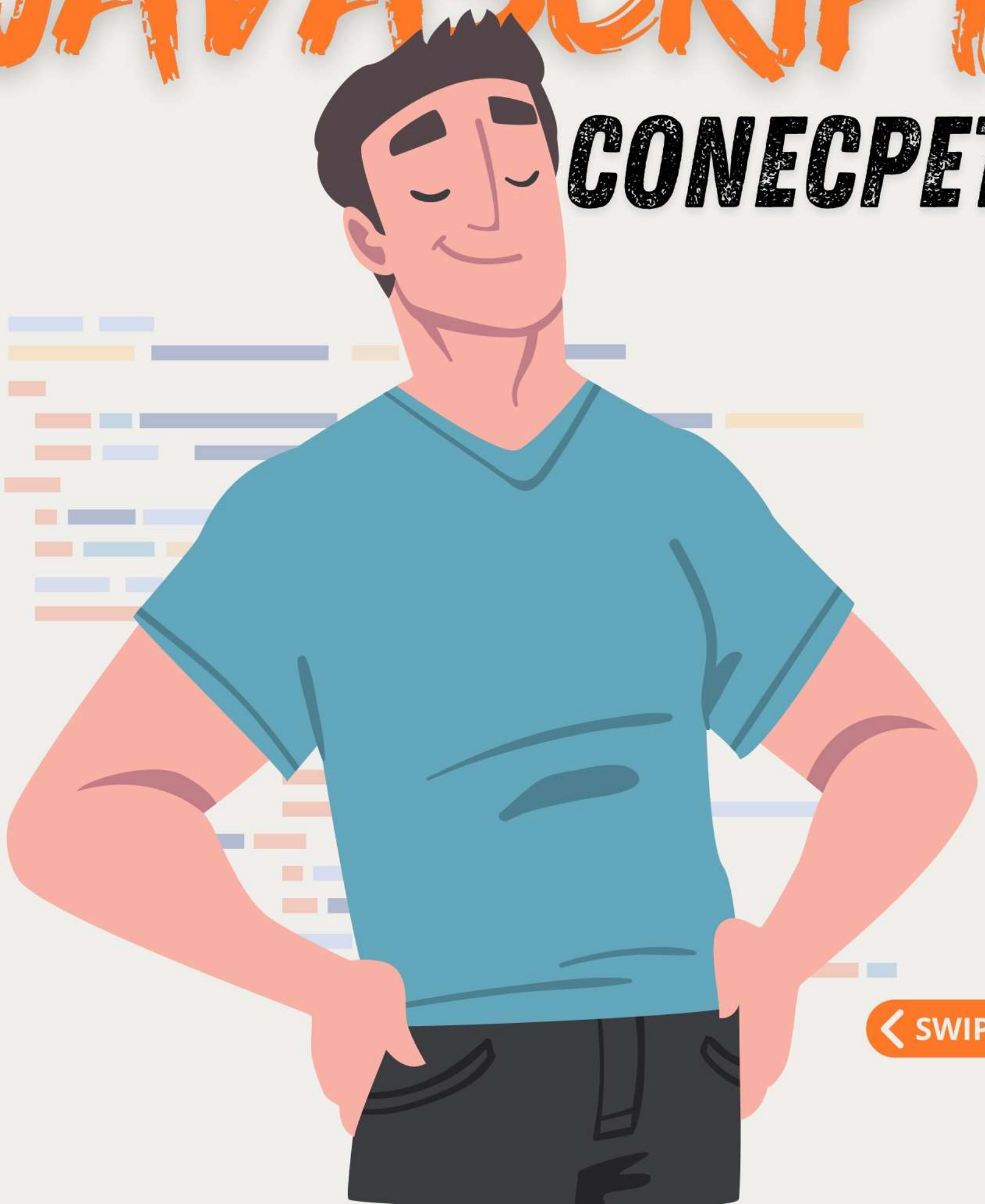
8 EXCITING NEW JAVASCRIPT CONECPETS



1/9

SWIPE

1. Optional Chaining (?.)

Introduced in ECMAScript 2020, optional chaining allows you to read the value of a property located deep within a chain of connected objects without having to check that each reference in the chain is valid.

OptionalChaining.js

```
let name = person?.address?.street?.name;
```


SWIPE

2. Nullish Coalescing (??)

Also introduced in ECMAScript 2020, the nullish coalescing operator returns the first operand if it's not null or undefined, and the second operand otherwise.

NullishCoalescing.js

```
let name = person?.name?? 'Unknown';
```


SWIPE

3. BigInt

A new numeric primitive in JavaScript, BigInt is used to represent integers with arbitrary precision, allowing for accurate calculations with large integers.

BigInt.js

```
const x = 12345678901234567890n;
```


SWIPE

4. globalThis

A new global object, globalThis, provides a way to access the global object in a way that's compatible with modern JavaScript environments.

globalThis.js

```
console.log(globalThis === window);
// true in a browser
```


SWIPE

5. matchAll()

A new method on the String prototype, matchAll() returns an iterator that yields matches of a regular expression against a string, including capturing groups.

matchAll.js

```
const regex = /(\w)(\d)/g;
const str = 'a1b2c3';
for (const match of str.matchAll(regex)) {
  console.log(match);
}
```


SWIPE

6. Promise.allSettled()

A new method on the Promise API, allSettled() returns a promise that is resolved when all of the promises in an array are either resolved or rejected.

promisteAllSettled.js

```
const promises = [Promise.resolve('a'), Promise.reject('b'), Promise.resolve('c')];
Promise.allSettled(promises).then((results) => console.log(results));
```


SWIPE

7. String.prototype.at()

A new method on the String prototype, at() returns the character at the specified index, allowing for negative indices to access characters from the end of the string.

stringPrototypeAt.js

```
const str = 'hello';
console.log(str.at(0)); // 'h'
console.log(str.at(-1)); // 'o'
```


SWIPE

8. Error Cause

A new property on Error objects, cause allows you to specify the underlying cause of an error.

errorCause.js

```
try {
  throw new Error("Error occurred", { cause: new Error("Underlying cause") });
} catch (error) {
  console.log(error.cause);
}
```


SWIPE