<script type="module"> import mermaid from 'https://cdnjs.cloudflare.com/ajax/libs/mermaid/10.4.0/mermaid.min.js'; mermaid.initialize({ startOnLoad: true }); </script>

Conditional Statements

- if-else
- switch
- ternary operator

```
if (condition) {
  // code block
} else if (condition) {
  // code block
} else {
```

```
switch (expression) {
  case x:
    // code block
    break;
  case y:
```

Loops and Iteration

- for
- while

```
for (initialize; test; increment) {
  // code block
}
```

```
while (condition) {
  // code block
}

do {
  // code block
} while (condition);
```

Jump Statements

statements.js

- break
- continue
- return
- throw

Miscellaneous Statements

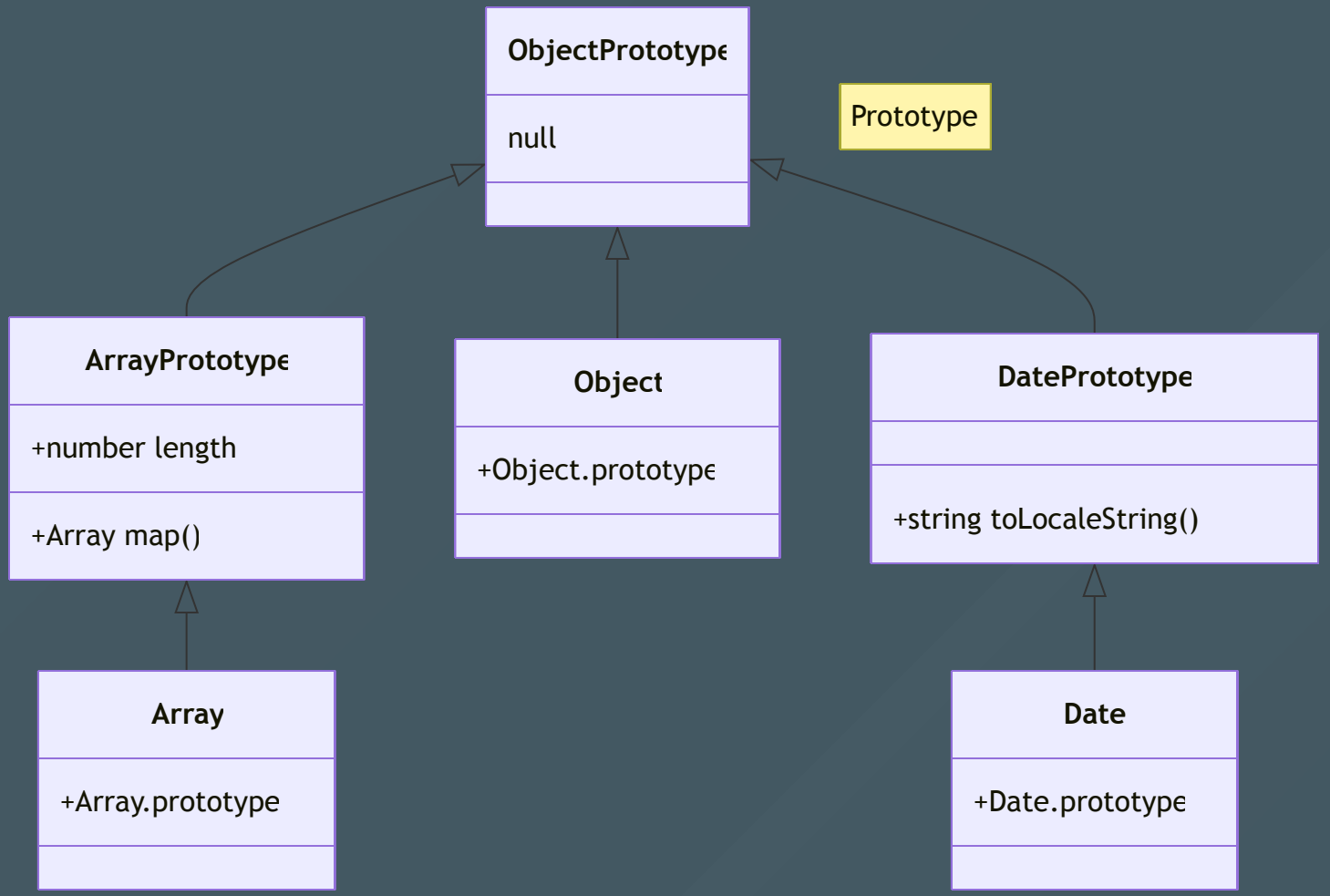
- debugger
- use strict
  - entire script or function, *not block*
  - all variables must be declared
  - functions invoked with `call` or `apply` must have a valid object for `this`
  - no duplicate property names in object literals
  - arguments object is not linked with parameters
  - ...

Objects

- creating objects
  - object literals
  - `new` keyword
  - `Object.create()`

Prototypes (easy version)

Every object has a second object (or null) associated with it, which is known as a prototype, and the first object inherits properties from the prototype.


ObjectPrototype
null
Prototype
ArrayPrototype
+number length
+Array map()
Object
+Object.prototype
DatePrototype
+string toLocaleString()
Array
+Array.prototype
Date
+Date.prototype

Querying and Setting Properties

- dot notation
- bracket notation

property access errors

```
addressLine1 = user.address.line1; // error if user.address is undefined
addressLine1 = user.address && user.address.line1; // safe
addressLine1 = user.address?.line1; // ES2020 // safe
```

Deleting Properties

- `delete` operator
- `Object.seal()` - can be changed but not added or deleted
- `Object.freeze()` - *make real constant objects*

Enumerating Properties

- `for...in`
- `Object.keys()`
- `Object.values()`
- `Object.entries()`

Getters and Setters

- `get` and `set` keywords
- `Object.defineProperty()`
- `Object.defineProperties()`

Serializing Objects

- `JSON.stringify()`
- `JSON.parse()`

Arrays

- creating arrays
- accessing array elements
- adding and removing elements
- iterating over arrays
- searching arrays

Creating Arrays

- array literals
- `new` keyword
  - `new Array()`
  - `new Array(num)`
  - `new Array(arg1, arg2, arg3, ...)`
- `Array.of()`
- `Array.from()`

Accessing Array Elements

- `arr[index]`
- `arr.length`
- `arr[arr.length - 1]`
- `arr[index] = 1`

```
var arr = [1, 2, 3];
arr[-1] = 10;
console.log(arr); // [1, 2, 3, -1: 10]

arr['1'] == arr[1]; // true
arr[1.0] == arr[1]; // true
```

Array Length

- `arr.length`
- `arr.length = 0`
- `arr.length = 10`

```
var arr = [1, 2, 3, 4, 5];

arr.length = 3;
console.log(arr); // [1, 2, 3]

arr.length = 0;
console.log(arr); // []

arr.length = 10;
console.log(arr); // [empty × 10]
```

Adding and Removing Elements

- `push()`
- `unshift()`
- `concat()`
- `splice()`

- `pop()`
- `shift()`
- `slice()`
- `splice()`

Iterating Over Arrays

- `for...in`
- `for...of`
- `forEach()`
- `map()`, `filter()`, `reduce()` (ES6)

Multi-dimensional Arrays

- `arr[row][column]`

```
var arr = [
  [1, 2, 3],
  [4, 5, 6]
];

console.log(arr[0][1]); // 2
```