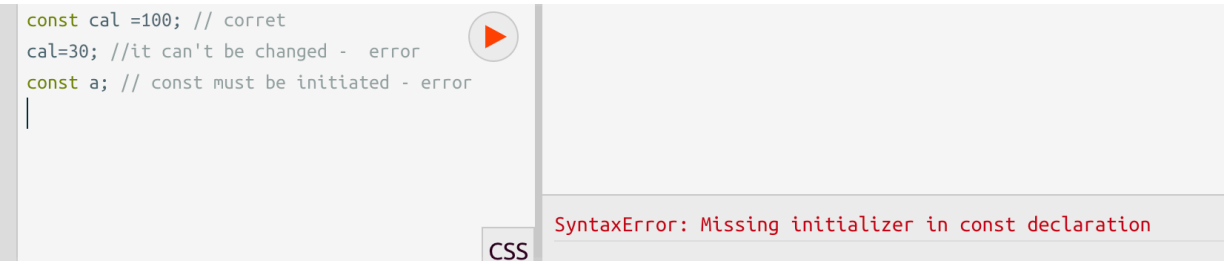
JavaScript Fundamentals
https://app.pluralsight.com/library/courses/javascript-fundamentals/table-of-contents

https://jscomplete.com/playground



Let and Var:

Var will call outside of block

```
if (true) {
    var foo = 9;
}
console.log(foo);   // 9
```

Let returns error when let call from outside block

```
if (true) {
    let foo = 9;
}
console.log(foo);   // error!
```

So Let is better than Var

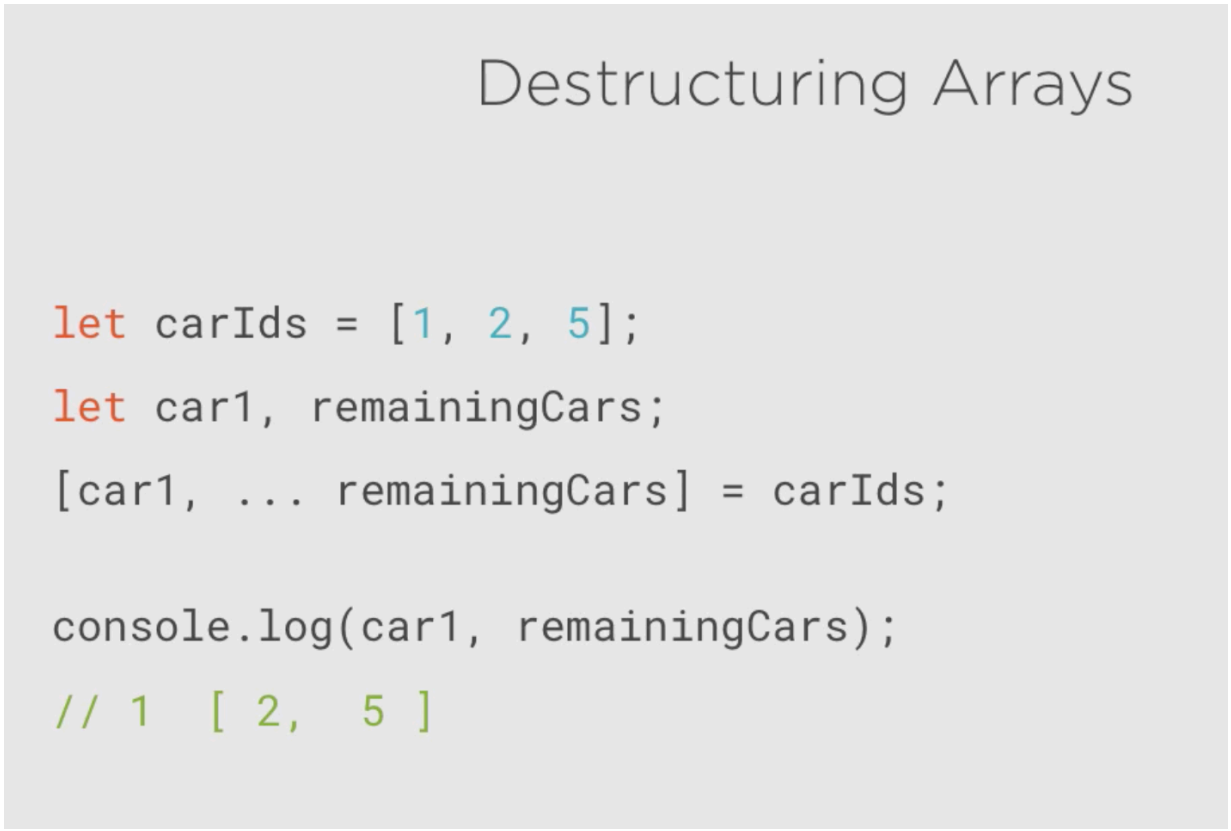
Rest Parameter:

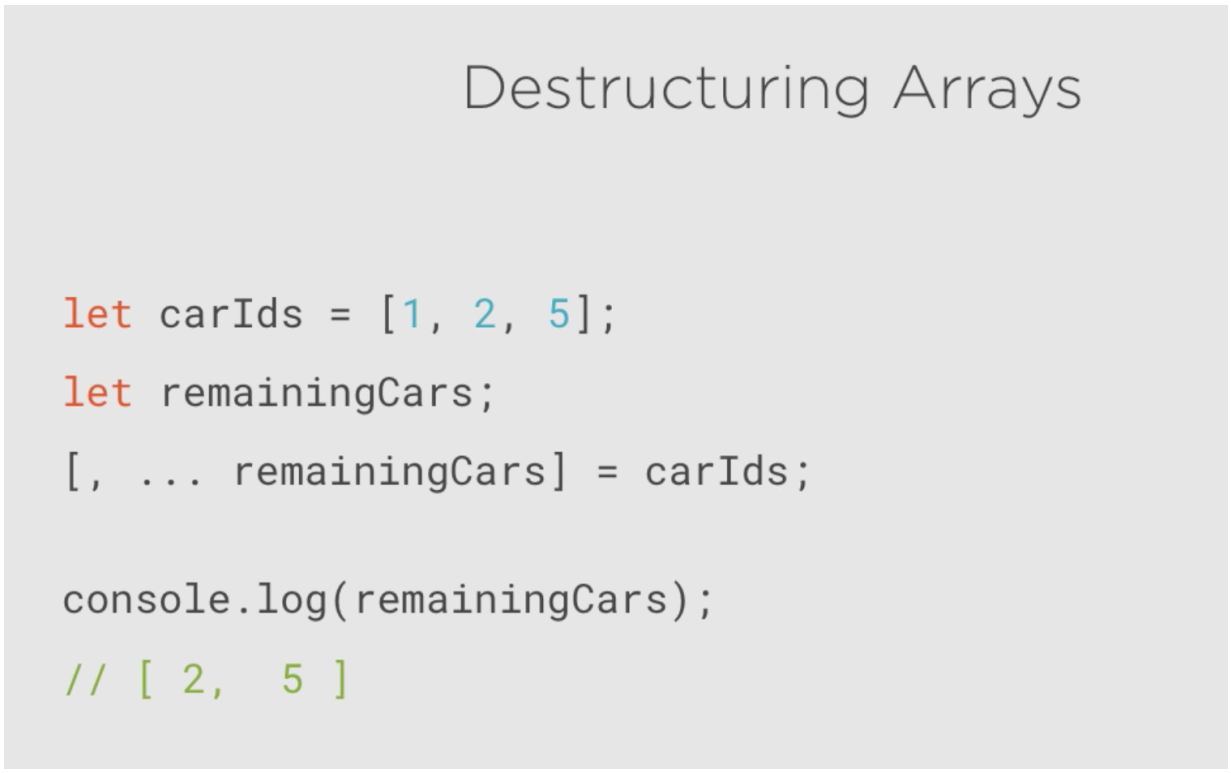
```
function sendCars(...allCarIds) {
    allCarIds.forEach( id => console.log(id));
}

sendCars(100, 200, 555);

// 100  200  555
```

Returns error. Because no comma allowed after "...carIds" (rest parameters)

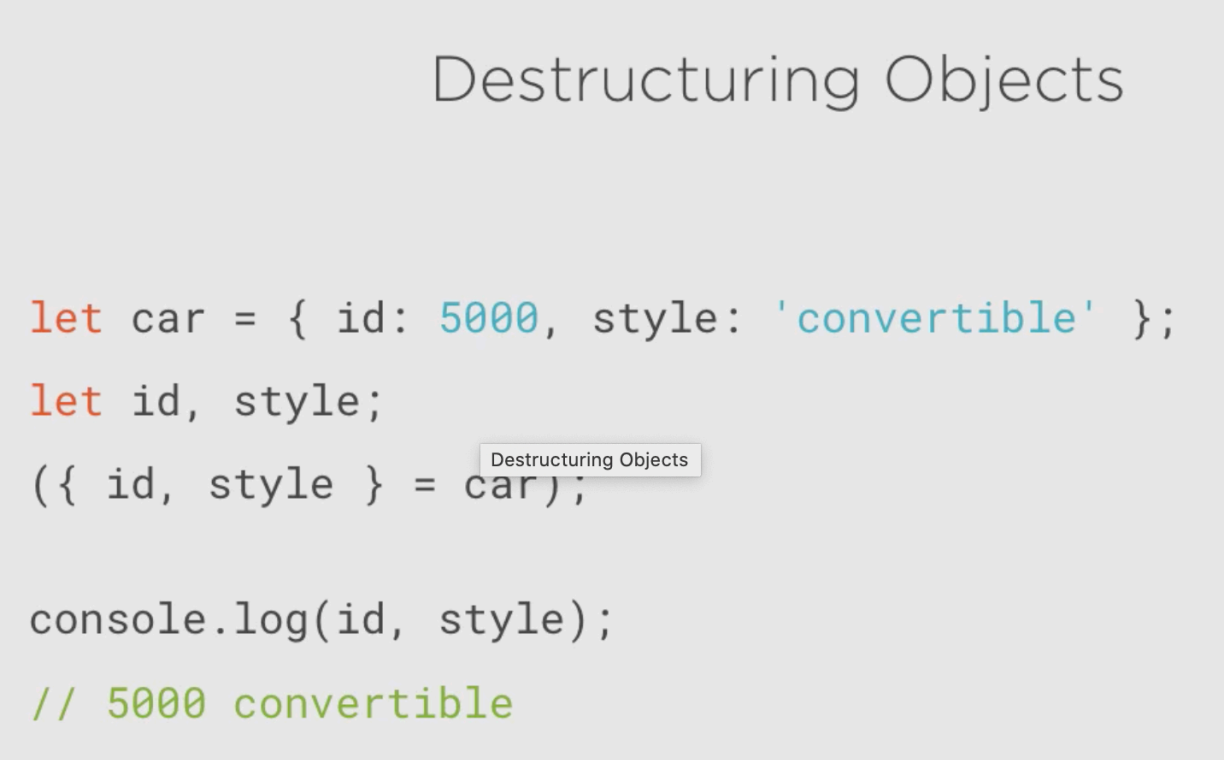
```
function sendCars(...carIds, day) {
  carIds.forEach( id => console.log(id) );
}
```

Destructuring Arrays:





Destructuring Objects:



Spread Syntax : opp to Rest parameters

Spread Syntax

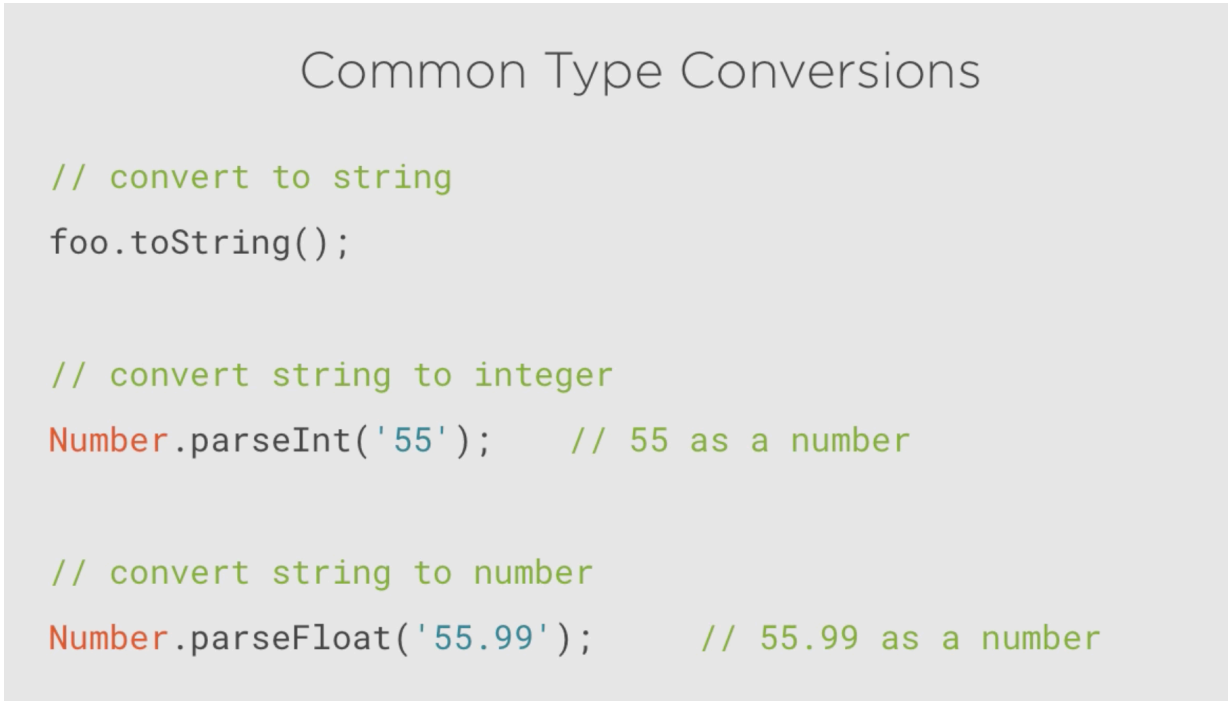
```
function startCars(car1, car2, car3) {
    console.log(car1, car2, car3);
}

let carIds = [100, 300, 500];
startCars(...carIds);
// 100  300  500
```

```
function startCars(car1, car2, car3) {
    console.log(car1, car2, car3);
}

let carCodes = 'abc';
startCars(...carCodes);
// a  b  c
```

TypeOF:

typeof()

```
typeof(1);    // number
typeof(true); // boolean
typeof('Hello'); // string
typeof( function(){} ); // function
typeof({}); // object
typeof(null); // object
typeof(undefined); // undefined
typeof(NaN); // number
```




```
console.log( typeof(Number.parseInt('55')));
```

```
console.log( Number.parseInt('55AB'));
```

returns 55

```
console.log( Number.parseFloat('55.88ABC'));
```

returns 55 .88

Functions & Nested Functions:

```
1
2   function startCar(carId) {
3     let message = 'Starting...';
4     let startFn = function turnKey() {
5       let message = 'Override';
6     };
7     startFn();
8     console.log(message);   // 'Starting...'
9   }
10  startCar(123);
11
12
13
```

Immediately invoked function expression:

```
(function() {
  console.log('in function');
})();
```

```
let app = (function() {
  let carId = 123;
  console.log('in function');
  return {};
})();

console.log(app);
```

Closures:

Call:

call

```
let o = {
    carId: 123,
    getId: function() {
        return this.carId;
    }
};

let newCar = { carId: 456 };

console.log( o.getId.call(newCar) ); // 456
```

Apply:

```
let o = {
    carId: 123,
    getId: function(prefix) {
        return prefix + this.carId;
    }
};

let newCar = { carId: 456 };

console.log( o.getId.apply(newCar, ['ID: ']) );
// ID: 456
```

Bind:

Arrow Function:

```
let getId = () => 123;

console.log( getId() );   // 123
```

() => 123 — Arrow function & No parameters to function
() — function
123 - return value
=> arrow

```
let getId = (prefix, suffix) => {
    return prefix + 123 + suffix;
};

console.log( getId('ID: ', '!') );  // ID: 123!
```

Constructor Function:

```
function Car(id) {
    this.carId = id;
    this.start = function() {
        console.log('start: ' + this.carId);
    };
}

let vehicle = new Car(123);
vehicle.start();
```

ProtoType:

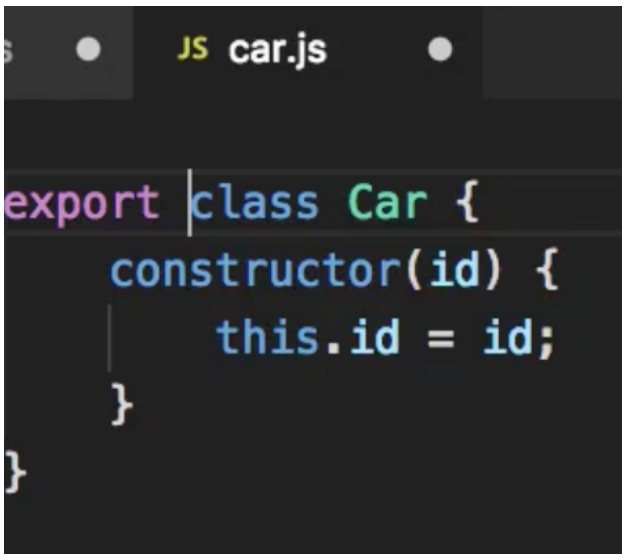
Prototypes

```
function Car(id) {
    this.carId = id;
}

Car.prototype.start = function() {
    console.log('start: ' + this.carId);
};

let car = new Car(123);
car.start();  // start: 123
```

Class, method

```
class Car {
    constructor(id) {
        this.id = id;
    }
    identify(suffix) {
        return `Car Id: ${this.id} ${suffix}`;
    }
}

let car = new Car(123);
console.log( car.identify('!!!') );
```

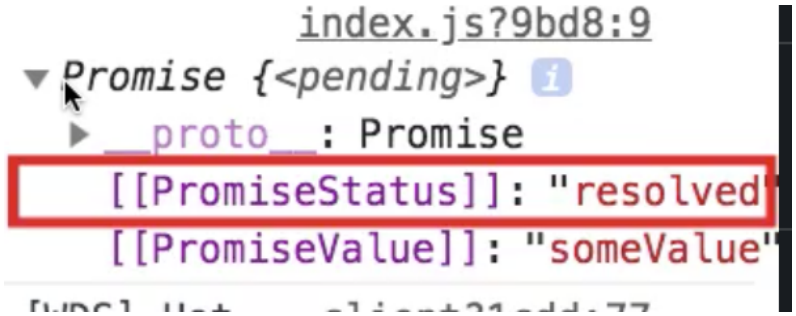
Creating Modules & importing modules:



```
import { Car } from './models/car.js';

let car = new Car(123);
console.log( car.id );
```

Create Promise:

```
let promise = new Promise(
    function(resolve, reject) {
        setTimeout(resolve, 100, 'someValue');
    }
);

console.log(promise);
```



```
let promise = new Promise(
    function(resolve, reject) {
        setTimeout(reject, 100, 'someValue');
    }
);

console.log(promise);
```

Settling a Promise

```
let promise = new Promise(
    function(resolve, reject) {
        setTimeout(resolve, 100, 'someValue');
    }
);
promise.then(
    value => console.log('fulfilled: ' + value),
    error => console.log('rejected: ' + error)
);
```