# Ecma 7 Cheat Sheet

### Exponentiation Operator

Performs exponential calculation on operands. Same algorithm as Math.pow(x, y).

```
let cubed = x => x ** 3;
cubed(2) //8;
```

### Async functions

Deferred generators.

```
function wait(t) {
    return new Promise((r) => setTim
}
async function asyncMania() {
    console.log("1");
    await wait(1000);
    console.log("2");
}
asyncMania()
.then(() => alert("3"));
//logs: 1 2 3
```

### Object Observe

Asynchronously observing the changes to an object.

```
var obj = {};
Object.observe(obj, function(changes
    console.log(changes);
});
obj.name = "hemanth";
//Would log -> [ { type: 'new`, obje
```

### Object.getOwnPropertyDescriptors

Returns a property descriptor for an own property.

```
//Creating a shallow copy.
var shallowCopy = Object.create(
    Object.getPrototypeOf(originalOb
    Object.getOwnPropertyDescriptors
);
```

### Object.values

Get all the values of the object as an array.

```
var person = { fname: "Hemanth", lna
Object.values(person);
```

### Object.entries

Returns a Array of arrays of key,value pairs.

```
var person = { fname: "Hemanth", lna
Object.entries(person);
//[["fname","Hemanth"],["lname","HM"
```

### Array.prototype.includes

Determines whether an array includes a certain element or not.

```
[1, 2, 3].includes(3, 0, 7); //true
[1, 2, NaN].includes(NaN); //true
[0,+1,-1].includes(42); //false
```

### Typed Objects

Portable, memory-safe, efficient, and structured access to contiguously allocated data.

```
var Point = new StructType({
    x: int32,
    y: int32
});
var point = new Point({
    x: 42,
    y: 420
});
```

### Trailing commas in function syntax

Trailing commas in parameter and argument lists.

```
var meow = function (cat1, cat2,) {}
Math.max(4,2,0,);
```

### Class properties

Properties of class.

```
class Cat {
    name = 'Garfield';
    static says = 'meow';
}
new Cat().name; //Garfield
Cat.says; //meow
```

### Map.prototype.toJSON

toJSON for Maps.

```
var myMap = new Map();
myMap.set(NaN, "not a number");
console.log(myMap.toJSON()); //{"NaN
```

### Set.prototype.toJSON

toJSON for Sets.

```
var mySet = new Set();
mySet.add(NaN);
mySet.add(1);
console.log(mySet.toJSON()) //{"1":1
```

### String.prototype.at

String containing the code point at the given position.

```
'abc𠮷def'.at(1) //'b'
'abc𠮷def'.at(3) //'𠮷'
```

### Object spread properties

Spread properties for object destructuring assignment.

```
let info = {fname, lname, ...rest};
info; //{ fname: "Hemanth", lname: "
```

### String.prototype.padLeft

left justify and pad strings.

```
"hello".padLeft(4); //"hello"
"hello".padLeft(20); //"hello"
"hello".padLeft(20, '1234') //"hello
```

### String.prototype.padRight

### String.prototype.trimLeft

### String.prototype.trimRight

Right justify and pad strings.

```
"hello".padRight(4); //"hello"
"hello".padRight(20; //hello"
"hello".padRight(20, '1234'); //"123
```

Left trim strings.

```
' \t \n LeftTrim  \t\n'.trimLeft();
//LeftTrim  \t\n
```

Right trim strings.

```
' \t \n TimeRight   \t\n'.trimLeft()
//\t \n LeftRight
```

### Regexp.escape

Escapes any characters that would have special meaning in a regular expression.

```
RegExp.escape("(*.*)"); //"(*.*)"
RegExp.escape("｡^･ɪ･^｡") //"｡^･ɪ･^｡"
RegExp.escape("😊 *_* +_+ ... 👍");
```

### Bind Operator

:: Function binding and method extraction

```
let log = (level, msg) => ::console[
import { map, takeWhile, forEach } f
getPlayers()
    ::map(x => x.character())
    ::takeWhile(x => x.strength > 10
    ::forEach(x => console.log(x));
```

### Reflect.Realm

TDB

```
TBD
```