ES6

VARIABLES

JavaScripts 6 (ES6) Variables

- Three different ways to declare variables
  ```
  var x = 123 // variable hoisting, can use before declaring
  let y = 234 // error if used before declaring
  const PI = 3.141593 // set value only at declaration
  ```

- Best practice
  - Declare variables at top of scope before using
  - Prefer **let** and **const** over **var**

JavaScript Object Notation (JSON)

```
let someObject = {
 anObjectProperty: {
  anotherObjectProp: {q: 111, w: 222},
  anotherArrayProp: [321, 432, 543]
 },
 aNumberArrayProp: [1, 2, 3],
 anObjectArrayProp: [
  {a: 123, b: 234},   {a: 321, b: 432}]}

console.log(someObject.anObjectProperty
    .anotherArrayProp[2]) // ==> 543
```

Variables can be scoped in code blocks

```
var i, x, a = [1, 2, 3]
for (i = 0;
    i < a.length;
    i++) {
 x = a[i];
}
```

ES5

```
let a = [1, 2, 3]
for (let i = 0;
    i < a.length;
    i++) {
 let x = a[i];
}
```

ES6

Scoped variables and closures

```
var callbacks = []
for (var i = 0; i <= 2; i++) {
 (function (i) {
   callbacks[i] = function() {
     return i * 2 }
 })(i);
}
callbacks[0]() === 0
callbacks[1]() === 2
callbacks[2]() === 4
```

```
let callbacks = []
for (let i = 0; i <= 2; i++) {

 callbacks[i] = function () {
     return i * 2 }
}

callbacks[0]() === 0
callbacks[1]() === 2
callbacks[2]() === 4
```

Variables can be scoped in code blocks

```
(function () {
 var foo = function ()
      { return 1 }
 foo() === 1
 (function () {
   var foo = function ()
      { return 2 }
   foo() === 2
 })();
 foo() === 1
})()
```

```
{
 function foo ()
      { return 1 }
 foo() === 1
 {
   function foo ()
      { return 2 }
   foo() === 2
 }
 foo() === 1
}
```

FUNCTIONS

Arrow Functions

ES5

```
function addEs5(a, b) {
 console.log(a, b);
 return a + b;
}
```

ES6

```
const addEs6 = (a, b) => {
 console.log(a, b);
 return a + b;
};
```

Single Line *Implied Return*

- ES5

```
function addEs5(a, b) {
 return a + b;
}
```

- ES6

```
const addEs6 = (a, b) => { return a + b; }
const addEs6 = (a, b) => a + b;
// return is optional if one line body return
```

Single Argument Optional Parens

- ES5

```
function squareEs5 (b) {
 return b * b
}
```

- ES6

```
const squareEs6 = b => b * b
// parenthesis is optional if one argument
```

This keyword

- ES5 variant 1

```
var self = this;
this.nums.forEach(function (v) {
 if (v % 5 === 0)
   self.fives.push(v);
});
```

- ES5 variant 2

```
this.nums.forEach(function (v) {
 if (v % 5 === 0)
   this.fives.push(v);
}, this);
```

- ES6 this behaves same as other modern languages

```
this.nums.forEach((v) => {
 if (v % 5 === 0)
   this.fives.push(v)
})
```

Default Parameters

```
const f = (x, y = 7, z = 42) => {
    return x + y + z;
}
f(1) === 50;
```