https://slides.com/concise/js/

concise JavaScript

A concise and accurate JavaScript tutorial/notes written for those entering the JavaScript world for the first time but already have experience with other languages

Some slides extracted from above reference

Definition

A method is *a* ***function***
as *some object's property*

The property which contains a value that
references to some function is called a “method.”

So is the referenced function.

Methods of An Object

```
// The cat object has three properties
// cat.age, cat.meow, and cat.sleep

var cat = {
    age: 3,
    meow: function () {}
};
cat.sleep = function () {};

// We would say that cat.meow and
// cat.sleep are "methods" of cat
```

Refer To The Object Inside A Method

When a function is invoked ***as a method*** of some object, the ***this*** value during the function call is *(usually)* bound to that object at ***run-time***

```
var cat = {
    age: 3,
    meow: function () {
        console.log(this.sound);
        return this.age;
    },
    sound: 'meow~~'
};

cat.meow(); // 3  ("meow~~" is printed)

var m = cat.meow;
m(); // TypeError or undefined
```

Methods

```
var cat = {
  age: 3,
  meow: function () {
    console.log(this.sound);
    return this.age;
  },
  sound: 'meow~~'
};

cat.meow();
```

Shorthand syntax for Methods

```
var cat = {
  age: 3,
  meow () {
    console.log(this.sound);
    return this.age;
  },
  sound: 'meow~~'
};

cat.meow();
```

Data Types in Javascript

Review The Data Types We've Seen So Far

Undefined *Null* *Boolean* *Number* *String* *Object*

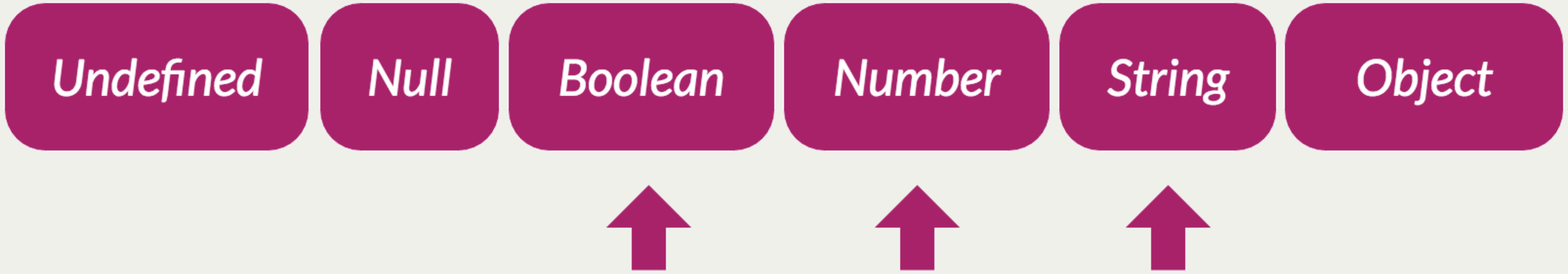
There are exactly ***6 types*** of ***values*** in JavaScript

Review The Data Types We've Seen So Far

Undefined *Null* *Boolean* *Number* *String* *Object*

These 2 are pretty boring

Review The Data Types We've Seen So Far



These 3 are more useful primitives

Review The Data Types We've Seen So Far

Undefined *Null* *Boolean* *Number* *String* *Object*

↑

This is the most interesting data type where we can start having ***nested*** and ***organized*** program ***structures***

"Object" Type Can Be Further *Categorized*

Undefined *Null* *Boolean* *Number* *String* *Object*

Object *Function* *Array*

Some Objects Are Called “Arrays”

Array Initialiser (Array Literal)

The notation using a pair of square brackets
to ***create/initialize*** a JavaScript ***Array*** object.

```
var w = [
    "test",
    1234,
    {},
    [],
    "hi"
];

w[4]; // "hi"
```

```
var w = new Array(5);
w[0] = "test";
w[1] = 1234;
w[2] = new Object();
w[3] = new Array();
w[4] = "hi";


w[4]; // "hi"
```

The code on the left-hand side has exactly the
same result as the one on the right-hand side

Enumerate All Elements In An Array (1/3)

There is a special property “length” for any Array object.

```
var arr = [ "test", 1234, {}, [], "hi" ];

for (var i = 0; i < arr.length; i += 1) {
    console.log(arr[i]);
}
```

NOTE: A “For-loop” is *not* always recommanded for enumerating all elements in an array, because...

Enumerate All Elements In An Array (2/3)

There is a special method “forEach” for any Array object.

```
var arr = [ "test", 1234, {}, [], "hi" ];

arr.forEach(function (val /*, i, arr*/) {
    console.log(val);
});
// undefined
```

The “***forEach***” method is much nicer...

Enumerate All Elements In An Array (3/3)

There is a special method “map” for any Array object.

```
var arr = [ "test", 1234, {}, [], "hi" ];

arr.map(function (val /*, i, arr*/) {
    return typeof val;
});
// [ "string",
//   "number",
//   "object",
//   "object",
//   "string" ]
```

We even have functional “*map*”, “*every*”, “*some*”, ... See the notes for more info

Append New Elements To An Array

There is a method “push” for all Array objects.
Or you can just assign a value to the corresponding slot.

```
var arr = [ "test", 1234, {}, [], "hi" ];

arr.push("sixth"); // 6
arr.length;        // 6
arr[5];            // "sixth"

arr[7] = 012;      // 10
arr.length;        // 8

arr[6];            // undefined
arr[7];            // 10

arr[8];            // undefined
arr.length;        // 8
```