Code Sandbox

Eloquent JavaScript Sandbox

JavaScript Terms

Array Methods
:

- `pop()` : remove last element
- `push()` : add an element at the end (could also use *a[a.length]=x*)
- `shift()` : removes first element of array
- `unshift()` : adds an element at beginning of array
- `spice(index, noremove, elements)` : adds elements to array at index after removing "noremove" elements. Example, `a.splice(0, 1)` removes the first element of an array.
- `concat()` : a.concat(b) joins b to a.
- `slice(start)` : slice an array into another array. slide returns an array (omitting `index` will return a copy of the array)
- `slice(start,end)` : slice an array from index to endindex (end not included) (MDN slice)
- `indexOf(value)` : finds the first index with a value of `value` [-1 if not found]
- `lastIndexOf(value)` : finds the last (searches from end) with a value of `value` [-1 if not found]
- `sort` : ()MDN Array sort)
- `map` : ()MDN Array map)
- `forEach` : ()MDN Array forEach)

Array Objects

- `delete a[0]` : deletes object in array at index 0. a[0] is now undefined.

Data Structures
: A variable that has not been assigned a value has the value undefined. (MDN Data Structures)
Flow Control
: MDN Flow Control

Functions - Parameters
: parameters are mutable. There is no way to make them constants StackOverflow JS Parameters

Immutable
: numbers, strings, and Booleans, are all immutable—it
is impossible to change values of those types. You *can* change the properties of object, causing a single
object value to have different content at different times.

You cannot add properties to strings

```
let kim = "Kim";
kim.age = 88;
console.log(kim.age);
// → undefined
```

Values of type string, number, and Boolean are not objects, and though
the language doesn't complain if you try to set new properties on them, it
doesn't actually store those properties. As mentioned earlier, such values are
immutable and cannot be changed.
But these types do have built-in properties.

Destructuring
: The destructuring assignment syntax is a JavaScript expression that makes it possible to unpack values from **arrays**, or properties from **objects**, into distinct variables.(Blog Site)
(MDN Site - Destructuring)

```
[a, b, ...rest] = [10, 20, 30, 40, 50];
const x = [1, 2, 3, 4, 5];
const [y, z] = x;
console.log(y); // 1
console.log(z); // 2

const foo = ['one', 'two', 'three'];
const [red, yellow, green] = foo;
console.log(red); // "one"
console.log(yellow); // "two"
console.log(green); // "three"

// more scenarios - see MDN article
```

Flow Control
: MDN Flow Control

Let & Const
: Even though number values don't change, you can use a let binding to keep track of a changing number by changing the value the binding points at. Similarly, though a const binding to an object can itself not be changed and will continue to point at the same object, the contents of that object might change.

```
const score = {visitors: 0, home: 0};
// This is okay
score.visitors = 1;
// This isn't allowed
score = {visitors: 1, home: 1};
```

Property
: A JavaScript property is a characteristic of an object, often describing attributes associated with a data structure. There are two kinds of properties: Instance properties hold data that are specific to a given object instance. Static properties hold data that are shared among all object instances.

Checking Property Programmatically
: use the `property name in object`

```
let anObject = {left: 1, right: 2};
console.log(anObject.left);
// → 1
delete anObject.left;
console.log(anObject.left);
// → undefined
console.log("left" in anObject);
// → false
console.log("right" in anObject);
// → true
```

Namespace
: JavaScript does not always warn you of using names already in use. The Math object provides a namespace for Math.min and Math.max and other mathematical functions. This makes it safe to use `let max = 5` or `const max = 5` in programs. JavaScript will warn on bindings using `let` or `const`. Javascript does *not* warn on bindings with `functions` and `var`.

Property
: Most values in JavaScript have properties which can be accessed by `value.prop` or `value["prop"]` (exception *null* and *undefined*). Objects use names for their properties. Arrays generally have numbers for their properties (starting from 0) There are some named properties in arrays such as `length` and a number of methods. Methods are functions that live in properties and (usually) act on
the value they are a property of. (Reference EloquentJS - Chapter 4 - Summary)

Pure function
: A pure function is a specific kind of value-producing function that not only has no side effects but also doesn't rely on side effects from other code—for example, it doesn't read global bindings whose value might change. A pure
function has the pleasant property that, when called with the same arguments, it always produces the same value (and doesn't do anything else). A call to such a function can be substituted by its return value without changing the
meaning of the code.

Rest Parameters - Functions
: The rest parameter syntax allows a function to accept an indefinite number of arguments as an array. (Wikipedia Variadic Functions) and (MDN Rest Parameter). The spread syntax can spread an array to help usage in functions with rest parameters (MDN Spread Parameter)

Serialize Data - JSON
: Method to save data for later, or, send to another system. All property names must be in double-quotes.
`JSON.stringify` : input is JavaScript, and output is JSON
`JSON.parse` : takes JSON and converts/encodes it to a value

String
: Strings: Every string value has a number of methods. Some very useful ones are slice and indexOf, which resemble the array methods of the same name. One difference is that a string's indexOf can search for a string containing
more than one character

- `indexOf()` : "I like popcorn".indexOf("like")
- `toUpperCase`
- `toLowerCase`
- `trim()` : method removes whitespace (spaces, newlines, tabs, and similar characters) from the start and end of a string.
- `padStart(length,padChar)` :

```
console.log(String(6).padStart(3, "0"));
// → 006
```

- `split(string)` : split on occurence of string (string is removed)
- `join(delimter)` : join, inserting delimeter between strings
- `repeat(number)` : repeat string number of times (creates a copy)

Tenary Operator
: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_Operator
isMember ? '$2.00' : '$10.00'

String Loops
: str.length, str.charAt(x), str[x], "for of" in ES6 `for (let letter of text)` , for …in `for (let i in str)`

```
// for ... of ...
for (let char of "Hello") {
  console.log(char);
}
```

Structs in JavaScript
: (StackOverflow Structs & JS)