

Default Parameters

- Functions can have default parameters
 - literal or computed by a function
 - **Example:** DefaultParameters.html

Rest Operator

- **Rest operator:** collects remaining items of iterable into an array
 - Uses a triple dot prefix (...x)
- **Rest parameters**
 - Rest operator appears at the end of the parameters list; it will receive all remaining parameters
 - Stores the remaining parameters as an array
 - **Example:** RestParameters.html

Spread Operator

- Opposite of rest operator
 - Converts items of an iterable (e.g., array) into arguments (for a function call) or into elements of array
 - Uses triple dot (exactly like rest operator)
 - Can appear anywhere (not just at the end)
 - Can be used inside array literals
- **Example:** SpreadOperator.html

Destructuring

- **Destructuring**

- A destructuring assignment allow us to tear apart an object or array
- **Example:** Destructuring.html

Arrow Functions

- Alternative to anonymous functions
 - “Lambda Expressions”
- Rely on the `=>` operator
- Format
 - Parameters `=>` code
 - Parenthesis for parameters are only required if the function has no parameters or 2 or more parameters. Function with one parameter do not require parenthesis surrounding the parameters
 - If code is a single expression no curly braces nor return statement are required
- **Example:** ArrowFunc.html

Symbols

- New primitive type in ECMAScript 6
- Serve as unique ids
- Special property keys
- Every symbol is unique
 - `Symbol() === Symbol()` is false
- Symbols can be used as property keys
 - Computed property key
 - Allows you to specify key of a property via an expression, by putting it in square brackets
- Optional string-valued parameter to provide a description
- Following operations ignore symbols
 - for-in loop
 - `Object.keys()`
 - `Object.getOwnPropertyNames()`
- Conversion of Symbol to Boolean returns true
- **Example:** `Symbol.html`

Sets

- **Set**

- Collection of keys. Keys can be primitive or references
- The Set constructor has zero or more arguments. With no arguments an empty Set is created
- If an argument is specified, it needs to be iterable (e.g., array)
- When iterating over sets, elements will be processed in the order they were inserted

- **Example:** Set.html

Map

- **Map**

- Collection of key / value pairs. Keys and values can be primitive or references
- Can define a map via iterable over key-value pairs
- `keys()` method returns iterable for keys in the map
- `values()` method returns iterable for values in the map
- `entries()` method returns iterable over (key, value) pairs

- **Example:** Map.html

WeakMaps

- **WeakMaps**

- Keys must be objects
- Key is weakly held; if the object representing a key is the only reference to the object, the object will be garbage-collected
- You cannot inspect the contents of a WeakMap
 - You cannot iterate over keys, values, or entries
 - You need a key to get some content out of the map

WeakMaps

- **Example:** WeakMap.html
 - As defined the example relies on a Map
 - Run the example in Chrome and using the **Memory** option run a “Take Heap Snapshot”
 - Search for “TerpObject” in the Constructor column
 - Repeat the above process, but using a WeakMap rather than a map
- Uses
 - To define private data in a “class”
- There are also WeakSets (set doesn’t prevent elements from being garbage-collected). Not that many use cases for them

References

- What Every JavaScript Developer Should Know About ECMAScript 2015 by K. Scott Allen
- <http://blog.teamtreehouse.com/reading-files-using-the-html5-filereader-api>
- <http://qanimate.com/post-series/ecmascript-6-complete-tutorial/>
- <https://spring.io/understanding/javascript-promises>
- <https://www.toptal.com/javascript/javascript-promises>
- <http://www.wintellect.com/devcenter/nstieglitz/5-great-features-in-es6-harmony>