

KYLE SIMPSON GETIFY@GMAIL.COM

JAVASCRIPT: THE RECENT PARTS

JavaScript Fatigue?

Declarative

Transpilers

Course Overview

JavaScript Versions

- ES6 / ES2015
- ES2016
- ES2017
- ES2018
- ES2019

Features

- Template Strings
- String padding/trimming
- Destructuring
- Array find() / includes()
- Array flat() / flatMap()
- Iterators, Generators
- RegExp Improvements
- **async .. await**
- **async* .. yield await**

...but before we begin...

Template Strings (Interpolated Literals)

"Interpoliterals"


```
1 var name = "Kyle Simpson";
2 var email = "getify@gmail.com";
3 var title = "Teacher";
4
5 var msg = "Welcome to this class! Your " +
6     title + " is " + name + ", contact: " +
7     email + ".";
8
9 // Welcome to this class! Your Teacher is
10 // Kyle Simpson, contact: getify@gmail.com.
```

string interpolation: imperative

```
1 var name = "Kyle Simpson";
2 var email = "getify@gmail.com";
3 var title = "Teacher";
4
5 var msg = `Welcome to this class! Your
6 ${title} is ${name}, contact: ${email}`;
7
8 // Welcome to this class! Your
9 // Teacher is Kyle Simpson, contact: getify@gmail.com.
```

string interpolation: declarative

```
1 var amount = 12.3;
2
3 var msg =
4     formatCurrency
5     `The total for your
6     order is ${amount}`;
7
8 // The total for your
9 // order is $12.30
```

string interpolation: tagged

```
1 function formatCurrency(strings, ...values) {
2     var str = "";
3     for (let i = 0; i < strings.length; i++) {
4         if (i > 0) {
5             if (typeof values[i-1] == "number") {
6                 str += `$$${values[i-1].toFixed(2)}`;
7             }
8             else {
9                 str += values[i-1];
10            }
11        }
12        str += strings[i]
13    }
14    return str;
15 }
```

string interpolation: tagged

```
1 function logger(strings, ...values) {
2   var str = "";
3   for (let i = 0; i < strings.length; i++) {
4     if (i > 0) {
5       if (values[i-1] && typeof values[i-1] == "object") {
6         if (values[i-1] instanceof Error) {
7           if (values[i-1].stack) {
8             str += values[i-1].stack;
9             continue;
10          }
11        }
12        else {
13          try {
14            str += JSON.stringify(values[i-1]);
15            continue;
16          }
17          catch (err) {}
18        }
19      }
20      str += values[i-1];
21    }
22    str += strings[i];
23  }
24  console.log(str);
25  return str;
26 }
```

interpolation: advanced

```
1 var v = 42;
2 var o = { a: 1, b: [2,3,4] };
3
4 logger`This is my value: ${v} and another: ${o}`;
5 // This is my value: 42 and another: {"a":1,"b":[2,3,4]}
6
7
8
9 try {
10     nothing();
11 }
12 catch (err) {
13     logger`Caught: ${err}`;
14 }
15 // Caught: ReferenceError: nothing is not defined
16 //   at <anonymous>:2:3
```

interpolation: advanced

String Padding

String Trimming

ES2017, ES2019


```
1 var str = "Hello";
2
3 str.padStart( 5 ); // "Hello"
4
5 str.padStart( 8 ); // "   Hello"
6
7 str.padStart( 8, "*" ); // "***Hello"
8
9 str.padStart( 8, "12345" ); // "123Hello"
10
11 str.padStart( 8, "ab" ); // "abaHello"
```

left start padding

```
1 var str = "Hello";
2
3 str.padEnd( 5 ); // "Hello"
4
5 str.padEnd( 8 ); // "Hello  "
6
7 str.padEnd( 8, "*" ); // "Hello***"
8
9 str.padEnd( 8, "12345" ); // "Hello123"
10
11 str.padEnd( 8, "ab" ); // "Helloaba"
```

right end padding

```
1 var str = "    some stuff  \t\t";
2
3 str.trim();           // "some stuff"
4
5 str.trimStart();     // "some stuff      "
6
7 str.trimEnd();       // "    some stuff"
```

left start trimming, right end trimming

Destructuring

decomposing a structure into
its individual parts

```
1  var tmp = getSomeRecords();
2
3  var first = tmp[0];
4  var second = tmp[1];
5
6  var firstName = first.name;
7  var firstEmail = first.email !== undefined ?
8      first.email :
9      "nobody@none.tld";
10
11 var secondName = second.name;
12 var secondEmail = second.email !== undefined ?
13     second.email :
14     "nobody@none.tld";
```

destructuring: imperative

```
1  var [
2      {
3          name: firstName,
4          email: firstEmail = "nobody@none.tld"
5      },
6      {
7          name: secondName,
8          email: secondEmail = "nobody@none.tld"
9      }
10 ] = getSomeRecords();
```

destructuring: declarative


```
1 function lookupRecord(store = "person-records", id = -1) {  
2     // ..  
3 }  
4  
5 function lookupRecord({  
6     store = "person-records",  
7     id = -1  
8 }) {  
9     // ..  
10 }  
11  
12 lookupRecord({id: 42});
```

destructuring: named arguments

```
1 // most common approach, using extend(..)
2
3 var defaults = {
4   url: "http://some.base.url/api",
5   method: "post",
6   headers: [
7     "Content-Type: text/plain"
8   ]
9 };
10
11 console.log(defaults);
12
13 // *****
14
15 var settings = {
16   url: "http://some.other.url/",
17   data: 42,
18   callback: function(resp) { /* .. */ }
19 };
20
21 // under core extend(..)
22 ajax( {_.extend({}, defaults, settings) } );
23
24 // or: ajax( Object.assign({}, defaults, settings) );
```

destructuring & restructuring

```
1 function ajaxOptions({
2     url = "http://some.base.url/api",
3     method = "post",
4     data,
5     callback,
6     headers: [
7         headers0 = "Content-Type: text/plain",
8         ...otherHeaders
9     ] = []
10 } = {}) {
11     return {
12         url, method, data, callback,
13         headers: [
14             headers0,
15             ...otherHeaders
16         ]
17     };
18 }
```

destructuring & restructuring

```
1 // with no arguments, returns the defaults
2 // as an object if necessary
3 var defaults = ajaxOptions();
4
5 console.log(defaults);
6
7 // *****
8
9 var settings = {
10     url: "http://some.other.url/",
11     data: 42,
12     callback: function(resp) { /* .. */ }
13 };
14
15 // with an argument, mixes in the settings
16 // w/ the defaults
17 ajax( ajaxOptions( settings ) );
```

destructuring & restructuring

Array .find(..)
Array .includes(..)

```
1 var arr = [ { a:1 }, { a:2 } ];
2
3 arr.find(function match(v){
4     return v && v.a > 1;
5 });
6 // { a:2 }
7
8 arr.find(function match(v){
9     return v && v.a > 10;
10 });
11 // undefined
12
13 arr.findIndex(function match(v){
14     return v && v.a > 10;
15 });
16 // -1
```

ES6: find / findIndex

```
1 var arr = [10,20,NaN,30,40,50];
2
3 arr.indexOf( 30 ) !== -1; // true
4
5 ~arr.indexOf( 20 ); // -2 (truthy)
6
7 ~arr.indexOf( NaN ); // -0 (falsy)
```

indexOf boolean hacking


```
1 var arr = [10,20,NaN,30,40,50];
2
3 arr.includes( 20 );           // true
4
5 arr.includes( 60 );           // false
6
7 arr.includes( 20, 3 );        // false
8
9 arr.includes( 10, -2 );       // false
10
11 arr.includes( 40, -2 );       // true
12
13 arr.includes( NaN );         // true
```

includes: API > syntax

Array .flat(..)
Array .flatMap(..)

```
1 var nestedValues =
2   [ 1, [2, 3], [[]], [4, [5] ], 6 ];
3
4 nestedValues.flat(0);
5 // [ 1, [2, 3], [[]], [4, [5] ], 6 ]
6
7 nestedValues.flat(/*default: 1*/);
8 // [ 1, 2, 3, [], 4, [ 5 ], 6 ]
9
10 nestedValues.flat(2);
11 // [ 1, 2, 3, 4, 5, 6 ]
```

Array: flat()

```
1 [1,2,3].map(function tuples(v){
2   return [ v * 2, String(v * 2) ];
3 });
4 // [ [2,"2"], [4,"4"], [6,"6"] ]
5
6 [1,2,3].map(function tuples(v){
7   return [ v * 2, String(v * 2) ];
8 }).flat();
9 // [ 2, "2", 4, "4", 6, "6" ]
10
11 [1,2,3].flatMap(function all(v){
12   return [ v * 2, String(v * 2) ];
13 });
14 // [ 2, "2", 4, "4", 6, "6" ]
```

Array: flatMap()

```
1 [1,2,3,4,5,6].flatMap(function doubleEvens(v){
2   if (v % 2 == 0) {
3     return [v, v * 2];
4   }
5   else {
6     return [];
7   }
8 });
9 // [ 2, 4, 4, 8, 6, 12 ]
```

Array: flatMap()

Iterators + Generators


```
1 var str = "Hello";
2 var world = ["W","o","r","l","d"];
3
4 var it1 = str[Symbol.iterator]();
5 var it2 = world[Symbol.iterator]();
6
7 it1.next(); // { value: "H", done: false }
8 it1.next(); // { value: "e", done: false }
9 it1.next(); // { value: "l", done: false }
10 it1.next(); // { value: "l", done: false }
11 it1.next(); // { value: "o", done: false }
12 it1.next(); // { value: undefined, done: true }
13
14 it2.next(); // { value: "W", done: false }
15 // ..
```

iterators: built-in iterators

```
1 var str = "Hello";
2
3 for (
4     let it = str[Symbol.iterator](), v, result;
5     (result = it.next()) && !result.done &&
6     (v = result.value || true);
7 ) {
8     console.log(v);
9 }
10 // "H" "e" "l" "l" "o"
```

iterators: imperative iteration

```
1 var str = "Hello";
2 var it = str[Symbol.iterator]();
3
4 for (let v of it) {
5     console.log(v);
6 }
7 // "H" "e" "l" "l" "o"
8
9 for (let v of str) {
10    console.log(v);
11 }
12 // "H" "e" "l" "l" "o"
```

iterators: declarative iteration

```
1 var str = "Hello";  
2  
3 var letters = [...str];  
4 letters;  
5 // ["H", "e", "\u0304", "\u0304", "o"]
```

iterators: declarative iteration

```
1  var obj = {
2      a: 1,
3      b: 2,
4      c: 3
5  };
6
7  for (let v of obj) {
8      console.log(v);
9  }
10 // TypeError!
```

iterators: objects not iterables

```
1  var obj = {
2      a: 1,
3      b: 2,
4      c: 3,
5      [Symbol.iterator]: function(){
6          var keys = Object.keys(this);
7          var index = 0;
8          return {
9              next: () =>
10                 (index < keys.length) ?
11                   { done: false, value: this[keys[index++]] } :
12                   { done: true, value: undefined }
13             };
14     }
15 };
16
17 [...obj];
18 // [1,2,3]
```

iterators: imperative iterator

```
1  function *main() {
2      yield 1;
3      yield 2;
4      yield 3;
5      return 4;
6  }
7
8  var it = main();
9
10 it.next();      // { value: 1, done: false }
11 it.next();      // { value: 2, done: false }
12 it.next();      // { value: 3, done: false }
13 it.next();      // { value: 4, done: true }
14
15 [...main()];
16 // [1,2,3]
```

iterators: generators

```
1  var obj = {
2      a: 1,
3      b: 2,
4      c: 3,
5      *[Symbol.iterator]() {
6          for (let key of Object.keys(this)) {
7              yield this[key];
8          }
9      }
10 };
11
12 [...obj];
13 // [1,2,3]
```

iterators: declarative iterator

RegExp Improvements

```
1 var msg = "Hello World";
2
3 msg.match(/(\.)/g);
4 // ["\.", "\."]
5
6 msg.match(/(\.)/g);
7 // ["\."]
8
9 msg.match(/(\.)(?=o)/g);
10 // ["\."]
11
12 msg.match(/(\.)(?!o)/g);
13 // ["\o", "\d"]
```

assertions, look ahead

```
1 var msg = "Hello World";
2
3 msg.match(/(?<=e)(\w+)/g);
4 // ["ll"]
5
6 msg.match(/(?<!e)(\w+)/g);
7 // ["lo", "ld"]
```

look behind

```
1 var msg = "Hello World";
2
3 msg.match(/.(l.)/);
4 // ["e1l","1l"]
5
6 msg.match(/([jkl])o Wor\1/);
7 // ["lo Worl","l"]
8
9 msg.match(/(?<cap>l.)/).groups;
10 // {cap: "1l"}
11
12 msg.match(/(?<cap>[jkl])o Wor\k<cap>/);
13 // ["lo Worl","l"]
14
15 msg.replace(/(?<cap>l.)/g, "-$<cap>-");
16 // "He-1l-o Wor-1d-"
17
18 msg.replace(/(?<cap>l.)/g, function re(...args) {
19     var [,,, { cap }] = args;
20     return cap.toUpperCase();
21 });
22 // "HeLLo WorLD"
```

named capture groups

```
1 var msg = `
2 The quick brown fox
3 jumps over the
4 lazy dog`;
5
6 msg.match(/brown.*over/);
7 // null
8
9 msg.match(/brown.*over/s);
10 // ["brown fox\njumps over"]
```

dotall /s

async .. await

```
1 fetchCurrentUser()
2 .then(function onUser(user){
3     return Promise.all([
4         fetchArchivedOrders( user.id ),
5         fetchCurrentOrders( user.id )
6     ]);
7 })
8 .then(function onOrders(
9     [ archivedOrders, currentOrders ]
10 ){
11     // ..
12 });
```

promise chains: yuck

```
1 runner(function *main(){
2     var user = yield fetchCurrentUser();
3
4     var [ archivedOrders, currentOrders ] =
5     yield Promise.all([
6         fetchArchivedOrders( user.id ),
7         fetchCurrentOrders( user.id )
8     ]);
9
10    // ..
11 });
```

sync-async (with generators)

```
1  async function main() {
2      var user = await fetchCurrentUser();
3
4      var [ archivedOrders, currentOrders ] =
5          await Promise.all([
6              fetchArchivedOrders( user.id ),
7              fetchCurrentOrders( user.id )
8          ]);
9
10     // ..
11 }
12
13 main();
```

async functions


```
1 async function fetchFiles(files) {  
2     var prs = files.map( getFile );  
3  
4     prs.forEach( function each(pr) {  
5         console.log( await pr );  
6     } );  
7 }
```

async FP iterations

github.com/getify/fasy

```
1  async function fetchFiles(files) {  
2      var prs = await FA.concurrent.map( getFile, files );  
3  
4      await FA.serial.forEach( async function each(pr){  
5          console.log( await pr );  
6      }, prs );  
7  }
```

fasy: **better** async FP iterations

- **await Only Promises**
- **Scheduling (Starvation)**
- **External Cancelation**

async functions: problems

github.com/getify/CAF

```
1 var token = new CAF.cancelToken();
2
3 var main = CAF( function *main(signal, url){
4     var resp = yield fetch( url, { signal } );
5     // ..
6
7     return resp;
8 } );
9
10 main( token.signal, "http://some.tld/other" )
11 .then( onResponse, onCancelOrError );
12
13 // only wait 5 seconds for the request!
14 setTimeout( function onElapsed(){
15     token.abort( "Request took too long!" );
16 }, 5000 );
```

cancelable async functions

```
1 var timeoutToken = CAF.timeout( 5000, "Took too long!" );
2
3 var main = CAF( function *main(signal,url){
4     var resp = yield fetch( url, { signal } );
5     // ..
6
7     return resp;
8 } );
9
10 main( timeoutToken "http://some.tld/other" )
11 .then( onResponse, onCancelOrError );
```

timeout cancelation

async* .. yield await

```
1  async function fetchURLs(urls) {
2      var results = [];
3
4      for (let url of urls) {
5          let resp = await fetch( url );
6          if (resp.status == 200) {
7              let text = await resp.text();
8              results.push( text.toUpperCase() );
9          }
10         else {
11             results.push( undefined );
12         }
13     }
14
15     return results;
16 }
```

async all-at-once

```
1  function *fetchURLs(urls) {
2      for (let url of urls) {
3          let resp = yield fetch( url );
4          if (resp.status == 200) {
5              let text = yield resp.text();
6              yield text.toUpperCase();
7          }
8          else {
9              yield undefined;
10         }
11     }
12 }
```

overloaded yield

```
1  async function *fetchURLs(urls) {
2      for (let url of urls) {
3          let resp = await fetch( url );
4          if (resp.status == 200) {
5              let text = await resp.text();
6              yield text.toUpperCase();
7          }
8          else {
9              yield undefined;
10         }
11     }
12 }
```

async generators

```
1  async function *fetchURLs(urls) {
2      var prs = urls.map(fetch);
3
4      for (let pr of prs) {
5          let resp = await pr;
6          if (resp.status == 200) {
7              let text = await resp.text();
8              yield text.toUpperCase();
9          }
10         else {
11             yield undefined;
12         }
13     }
14 }
```

async generators: upfront


```
1 for (let text of fetchURLs( favoriteSites )) {  
2   console.log( text );  
3 }
```

```
1 var it = fetchURLs( favoriteSites );  
2  
3 while (true) {  
4   let res = it.next();  
5   if (res.done) break;  
6   let text = res.value;  
7  
8   console.log( text );  
9 }
```

async iteration: busted

```
1 async function main(favoriteSites) {
2     var it = fetchURLs( favoriteSites );
3
4     while (true) {
5         let res = await it.next();
6         if (res.done) break;
7         let text = res.value;
8
9         console.log( text );
10    }
11 }
```

```
1 async function main(favoriteSites) {
2     for await (let text of fetchURLs( favoriteSites )) {
3         console.log( text );
4     }
5 }
```

async iteration: hooray!

**"Hang on tight and
enjoy the ride!"**

THANKS!!!!

KYLE SIMPSON GETIFY@GMAIL.COM

**JAVASCRIPT:
THE RECENT PARTS**