Rest / spread

- Vi har pratat om rest-parametrar:
- `function (…numbers) {}`
- Det finns en väldigt lik operator som heter spread.
- Spread syntax (...) allows an iterable such as an array expression or string to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected, or an object expression to be expanded in places where zero or more key-value pairs (for object literals) are expected.
- https://developer.mozilla.org/sv-SE/docs/Web/JavaScript/Reference/Operators/Spread_syntax

Spread

```
function sum(x, y, z) {
  return x + y + z;
}

const numbers = [1, 2, 3];

console.log(sum(...numbers));
// expected output: 6
```

Rekursiva funktioner

- Rekursiva funktioner är funktioner som anropar sig själva.
- Factorial - the product of an integer and all the integers below it; e.g. factorial four (4!) is equal to 24.

```
function factorial(n) {
  if ((n === 0) || (n === 1))
    return 1;
  else
    return (n * factorial(n - 1));
}
```

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Functions#Recursion

Reguljära uttryck

- Regular expressions are patterns used to match character combinations in strings. In JavaScript, regular expressions are also objects.
- 1)

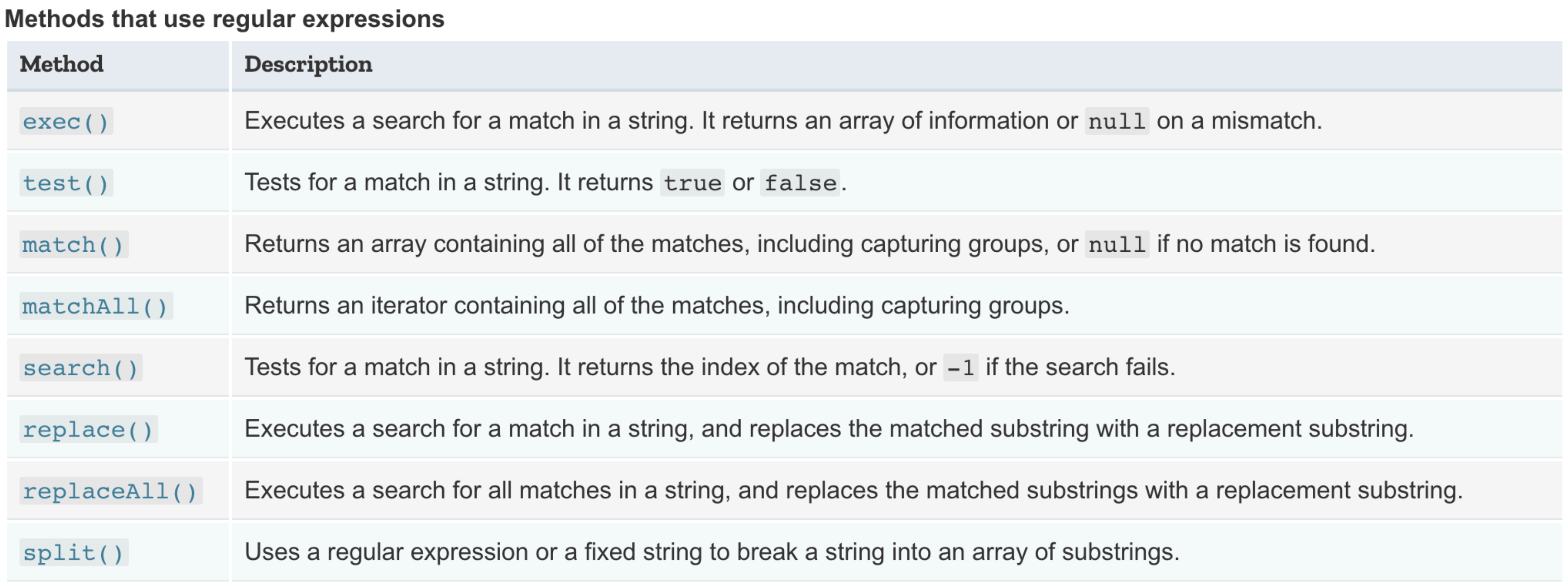
```
let re = /ab+c/;
```

- 2)

```
let re = new RegExp('ab+c');
```

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions

Reguljära uttryck



Methods that use regular expressions

Method	Description
`exec()`	Executes a search for a match in a string. It returns an array of information or `null` on a mismatch.
`test()`	Tests for a match in a string. It returns `true` or `false`.
`match()`	Returns an array containing all of the matches, including capturing groups, or `null` if no match is found.
`matchAll()`	Returns an iterator containing all of the matches, including capturing groups.
`search()`	Tests for a match in a string. It returns the index of the match, or `-1` if the search fails.
`replace()`	Executes a search for a match in a string, and replaces the matched substring with a replacement substring.
`replaceAll()`	Executes a search for all matches in a string, and replaces the matched substrings with a replacement substring.
`split()`	Uses a regular expression or a fixed string to break a string into an array of substrings.

Template literals
String interpolation

```
let name = 'Bob', time = 'today';
'Hello ' + name + ', how are you ' + time + '?'
```

```
let name = 'Bob', time = 'today';
`Hello ${name}, how are you ${time}?`
```

Special characters

```
'one line \n another line'
```

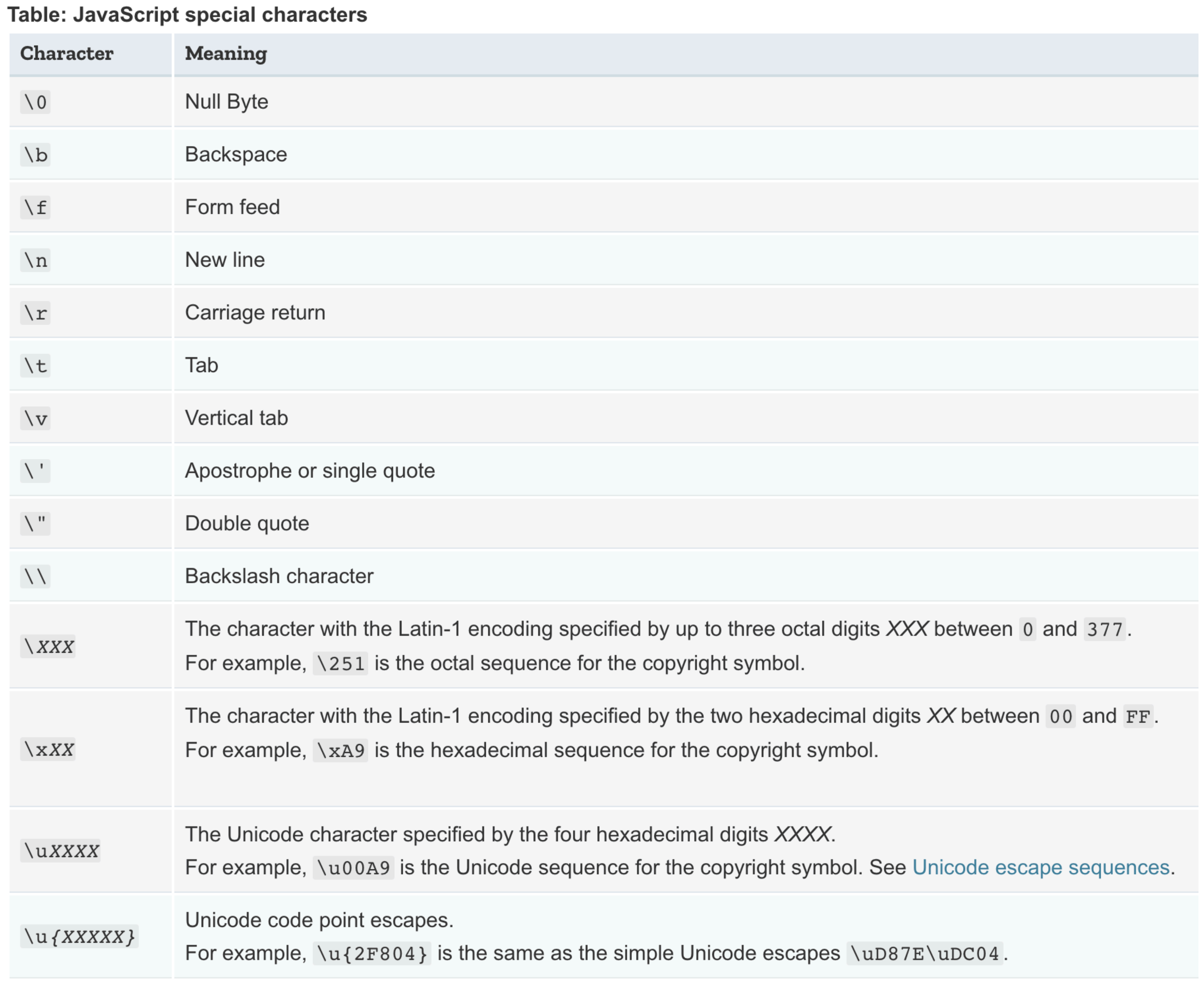


Table: JavaScript special characters

Character	Meaning
`\0`	Null Byte
`\b`	Backspace
`\f`	Form feed
`\n`	New line
`\r`	Carriage return
`\t`	Tab
`\v`	Vertical tab
`\'`	Apostrophe or single quote
`\"`	Double quote
`\\`	Backslash character
`\XXX`	The character with the Latin-1 encoding specified by up to three octal digits *XXX* between `0` and `377`. For example, `\251` is the octal sequence for the copyright symbol.
`\xXX`	The character with the Latin-1 encoding specified by the two hexadecimal digits *XX* between `00` and `FF`. For example, `\xA9` is the hexadecimal sequence for the copyright symbol.
`\uXXXX`	The Unicode character specified by the four hexadecimal digits *XXXX*. For example, `\u00A9` is the Unicode sequence for the copyright symbol. See Unicode escape sequences.
`\u{XXXXX}`	Unicode code point escapes. For example, `\u{2F804}` is the same as the simple Unicode escapes `\uD87E\uDC04`.

Throw exception

- Om ett fel uppstår kan vi kasta fel eller exceptions.

```
throw new Error('Whoops!')
```

```
throw 'Error2';    // String type
throw 42;          // Number type
throw true;        // Boolean type
throw {toString: function() { return "I'm an object!"; } };
```

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Error

Try… catch

- För att hantera ett Error eller en funktion som kan kasta ett Error kan man använda try… catch för att fånga det.

```
try {
  throw new Error('Whoops!')
} catch (e) {
  console.error(e.name + ': ' + e.message)
}
```

Try... catch

```
function getMonthName(mo) {
  mo = mo - 1; // Adjust month number for array index (1 = Jan, 12 = Dec)
  let months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul',
                'Aug', 'Sep', 'Oct', 'Nov', 'Dec'];
  if (months[mo]) {
    return months[mo];
  } else {
    throw 'InvalidMonthNo'; // throw keyword is used here
  }
}

try { // statements to try
  monthName = getMonthName(myMonth); // function could throw exception
}
catch (e) {
  monthName = 'unknown';
  logMyErrors(e); // pass exception object to error handler (i.e. your
own function)
}
```

Hantera olika fel

```
try {
  foo.bar()
} catch (e) {
  if (e instanceof EvalError) {
    console.error(e.name + ': ' + e.message)
  } else if (e instanceof RangeError) {
    console.error(e.name + ': ' + e.message)
  }
  // ... etc
}
```

Finally

- The `finally` block contains statements to be executed after the `try` and `catch` blocks execute.

```
openMyFile();
try {
  writeMyFile(theData); // This may throw an error
} catch(e) {
  handleError(e); // If an error occurred, handle it
} finally {
  closeMyFile(); // Always close the resource
}
```

Symbols

- ECMAScript 6 introduces a new primitive type: symbols.
- Tokens that serve as unique IDs.
- Create symbols via the factory function Symbol() (loosely similar to String returning strings if called as a function)

```
let symbol1 = Symbol();
```

- https://developer.mozilla.org/en-US/docs/Glossary/Symbol
- https://2ality.com/2014/12/es6-symbols.html
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Symbol

Maps

- ECMAScript 2015 introduces a new data structure to map values to values.
- A Map object is a simple key/value map and can iterate its elements in insertion order.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Keyed_collections

Maps

```
let sayings = new Map();
sayings.set('dog', 'woof');
sayings.set('cat', 'meow');
sayings.set('elephant', 'toot');
sayings.size; // 3
sayings.get('dog'); // woof
sayings.get('fox'); // undefined
sayings.has('bird'); // false
sayings.delete('dog');
sayings.has('dog'); // false

for (let [key, value] of sayings) {
  console.log(key + ' goes ' + value);
}
// "cat goes meow"
// "elephant goes toot"

sayings.clear();
sayings.size; // 0
```

Object.defineProperty()

- The static method Object.defineProperty() defines a new property directly on an object, or modifies an existing property on an object, and returns the object.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty

Object.preventExtensions()

- The Object.preventExtensions() method prevents new properties from ever being added to an object (i.e. prevents future extensions to the object).
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/preventExtensions

Strict mode

- Strict mode makes several changes to normal JavaScript semantics:
  1. Eliminates some JavaScript silent errors by changing them to throw errors.
  2. Fixes mistakes that make it difficult for JavaScript engines to perform optimizations: strict mode code can sometimes be made to run faster than identical code that's not strict mode.
  3. Prohibits some syntax likely to be defined in future versions of ECMAScript.
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Strict_mode

Strict mode

```
'use strict';

// Assignment to a non-writable global
var undefined = 5; // throws a TypeError
var Infinity = 5; // throws a TypeError

// Assignment to a non-writable property
var obj1 = {};
Object.defineProperty(obj1, 'x', { value: 42, writable: false });
obj1.x = 9; // throws a TypeError

// Assignment to a getter-only property
var obj2 = { get x() { return 17; } };
obj2.x = 5; // throws a TypeError

// Assignment to a new property on a non-extensible object
var fixed = {};
Object.preventExtensions(fixed);
fixed.newProp = 'ohai'; // throws a TypeError
```