JESUÏTES educació

DAWM06UF2

ESTRUCTURES DEFINIDES PEL PROGRAMADOR

OBJECTES

# UF2.5 ESTRUCTURES DEFINIDES PEL PROGRAMADOR. EXPRESSIONS REGULARS

CFGS Desenvolupament d'Aplicacions Web

**M06. Desenvolupament web en entorn client**

Fundació Jesuïtes Educació - Escola del Clot

Sergi Grau sergi.grau@fje.edu

# OBJECTIUS



+ Aprendre el funcionament de les Expressions regulars

# INTRODUCCIÓ

- Les expressions regulars són patrons que s'utilitzen per fer coincidir combinacions de caràcters en cadenes.

- En JavaScript, les expressions regulars **també són objectes**.

- Aquests patrons s'utilitzen amb els mètodes exec () i test () de RegExp, i amb match (), matchAll (), replace (), replaceAll (), search () i split () mètodes de String.

# CREAR UNA EXPRESSIÓ REGULAR



- Usant una expressió regular literal, que consisteix en un patró tancat entre barres, com segueix:

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

- Les expressions regulars literals proporcionen la compilació de l'expressió regular quan es carrega l'script. Si l'expressió regular roman constant, el seu ús pot millorar el rendiment.

- O cridant a la funció constructora d'l'objecte RegExp, de la següent manera. Fes servir la funció constructora quan sàpigues que el patró de l'expressió regular canviarà, o no coneixes el patró i el obtens d'una altra font, com l'entrada de l'usuari.

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

# ESCRIURE UN PATRÓ DE EXPRESSIÓ REGULAR



- Un patró d'expressió regular es compon de caràcters simples, com /abc /, o una combinació de caràcters simples i especials, com /b*c/ o /Capítol(\ d) \.\ D*/.

- L'últim exemple inclou parèntesi, que s'utilitzen com a dispositius de memòria. La coincidència realitzada amb aquesta part de el patró es recorda per al seu ús posterior,

# UTILITZA PATRONS SIMPLES

- Els patrons simples es construeixen amb caràcters per als que vols trobar una coincidència directa. Per exemple, el patró /abc/ coincideix amb combinacions de caràcters en cadenes només quan ocorre la seqüència exacta "abc" (tots els caràcters junts i en aquest ordre).

- Tal coincidència tindria èxit en les cadenes "Hola, coneixes el teu abc?" i "Els últims dissenys d'avions van evolucionar a partir d'slabcraft". En tots dos casos, la coincidència és amb la subcadena "abc". No hi ha cap coincidència en la cadena "Grab crab" perquè encara que conté la subcadena "ab c", no conté la subcadena "abc" exacta.

# UTILITZA PATRONS SIMPLES

+ Els patrons simples es construeixen amb caràcters per als que vols trobar una coincidència directa. Per exemple, el patró /abc/ coincideix amb combinacions de caràcters en cadenes només quan ocorre la seqüència exacta "abc" (tots els caràcters junts i en aquest ordre).

+ Tal coincidència tindria èxit en les cadenes "Hola, coneixes el teu abc?" i "Els últims dissenys d'avions van evolucionar a partir d'slabcraft". En tots dos casos, la coincidència és amb la subcadena "abc". No hi ha cap coincidència en la cadena "Grab crab" perquè encara que conté la subcadena "ab c", no conté la subcadena "abc" exacta.

# UTILITZA CARÀCTERS ESPECIALS

+ Quan la recerca d'una coincidència requereix alguna cosa més que una coincidència exacta, com ara buscar una o més 'b', o trobar espais en blanc, pots incloure caràcters especials en el patró. Per exemple, per fer coincidir una sola "a"seguida de zero o més "b"s seguides de"c" , faries servir el patró /ab*c/: el *després de "b"dir "0 o més aparicions de l'element anterior". A la cadena "cbbabbbbcdebc", aquest patró coincidirà amb la subcadena "abbbbc".

# ÚS BÀSIC

- / regex / flags, i.e /[A-Z]+/g
- / DAW2\?*\\/  s’escapa amb \
- () grups en parèntisi
- | or lògic

# MODIFICADORS BÀSICS

JESUÏTES educació
DAWM06UF4 sergi.grau@fje.edu

JS

```
//exemples d'expressions regulars
//la sintaxi és /pattern/modifiers;
let expressio = /daw2/i; // i no és sensible al cas.
let n = "i like DAW2".search(expressio);
console.log(n); // posicio en que apareix
let cadena = "i like daw2".replace(expressio, 'DAM2');
console.log(cadena); // cadena reemplaçada
expressio = /daw2/ig; // no s'atura en la primera ocurrència
cadena = "a DAW2 l'agrafa daw2".replace(expressio, 'DAM2');
console.log(cadena); // cadena reemplaçada

expressio = /daw2/img; //es per quan tenim vàries linies
trobades = "a DAW2 l'agrafa daw2".match(expressio);
console.log(trobades);


  7
  i like DAM2
  a DAM2 l'agrafa DAM2
  [ 'DAW2', 'daw2' ]
```

JS

```
//exemples d'ús de caràcters

let cadena = "javascript avançant?";
let expressio= /[a]/g;
let trobades = cadena.match(expressio);
console.log(trobades);
```

```
[ 'a', 'a', 'a', 'a', 'a' ]
```

```
let cadena = "javascript avançant?”;
let expressio= /[a-c]/g;
let trobades = cadena.match(expressio);
console.log(trobades);
```

```
[ 'a', 'a', 'c', 'a', 'a', 'a' ]
```

JS

```
//exemples d'ús de caràcters

let cadena = "javascript avançant?";
let expressio= /[a]/g;
let trobades = cadena.match(expressio);
console.log(trobades);


[ 'a', 'a', 'a', 'a', 'a' ]



let cadena = "javascript avançant?”;
let expressio= /[a-c]/g;
let trobades = cadena.match(expressio);
console.log(trobades);

[ 'a', 'a', 'c', 'a', 'a', 'a' ]


let expressio= /(dam2|daw1|asix2)/g;
let cadena = "després de daw1 o dam2 o asix2 fem daw2".replace(expressio, '---');
console.log(cadena);


després de --- o --- o --- fem daw2
```

# METACARÀCTERS



JS

```
let expressio= /\D/g; //d correson a un digit, D implica que no és un dígit
let cadena = "després de daw1 o dam2 o asix2 fem daw2".replace(expressio, 'X');
console.log(cadena);

després de dawX o damX o asixX fem dawX
XXXXXXXXXXXXX1XXXXXX2XXXXXXX2XXXXXXXX2

let expressio= /\s/g; //s correson a un espai en blanc, S implica que no és un espai en blanc
let cadena = "després de daw1 o dam2 o asix2 fem daw2".replace(expressio, '_');
console.log(cadena);

després_de_daw1_o_dam2_o_asix2_fem_daw2
_______ __ _____ _ _____ _ ______ ___ _____


let expressio= /\bes/g; //b a inici de paraula
let pos = "es o mes".search(expressio);
console.log(pos);
0
```

# QUANTIFICADORS



JS

```
let cadena = "jaaaava no es javaascript!";
let expressio = /aa+/g; //+ 1 o més ocurrències  //* 0 o més ocurrències //? 1 o 0 aparicions
let resultat = cadena.match(expressio);
console.log(resultat);
```

```
[ 'aaaa', 'aa' ]
```

```
[ 'aaaa', 'a', 'a', 'aa' ]
```

```
console.log(/e/.test('javascript'));

false
```

# CLASSES DE CARÀCTER



- \w word \d digit \s whitespace (tabs, line breaks)
- \W NOT word \D NOT digit \S NOT whitespace
- \t tabs, \n line breaks
- . qualsevol caràcter (excepte nova linia)
- [xyz] coincideix amb qualsevol x, y, z
- [J-Z] coincideix amb qualsevol entre J i Z.
- [^xyz] NO x, y, z

# QUANTIFICACIÓ

- bob|alice coincideix amb bob o alice
- z? zero o una ocurrències
- z* zero i múltiples ocurrències
- z+ una o múltiples ocurrències
- z{n} n ocurrències
- z{min,max} min/max ocurrències

+ hello world coincidència exacta

+ ^hello comença per

+ world$ acaba per

```
+ const matches = 'aBC'.match(/[A-Z]/g);
+ // sortida: Array [B, C]

+ const index = 'aBC'.search(/[A-Z]/);
+ // sortida: 1

+ const next = 'aBC'.replace(/a/, 'A');
+ // sortida: ABC
```

# CARÀCTERS ESPECIALS EN EXPRESSIONS REGULARS.

JESUÏTES educació
DAWM06UF2 sergi.grau@fje.edu

Caràcters especials en expressions regulars.

| Caràcters / construccions | article corresponent |
|---|---|
| `\, ., \cX, \d, \D, \f, \n, \r, \s, \S, \t, \v, \w, \W, \0, \xhh, \uhhhh, \uhhhhh, [\b]` | Classes de caràcters |
| `^, $, x(?=y), x(?!y), (?<=y)x, (?<!y)x, \b, \B` | assercions |
| `(x), (?:x), (?<Name>x), x\|y, [xyz], [^xyz], \Number` | Grups i rangs |
| `*, +, ?, , , x{n} x{n,} x{n,m}` | quantificadors |
| `\p{UnicodeProperty}, \P{UnicodeProperty}` | Fuites de propietats Unicode |

Mètodes que fan servir expressions regulars

| mètode | Descripció |
| --- | --- |
| `exec()` | Executa una recerca per una coincidència en una cadena. Retorna un arranjament d'informació o `null` en una discrepància. |
| `test()` | Prova una coincidència en una cadena. Retorna `true` o `false`. |
| `match()` | Retorna un arranjament que conté totes les coincidències, inclosos els grups de captura, o `null` si no es troba cap coincidència. |
| `matchAll()` | Retorna un iterador que conté totes les coincidències, inclosos els grups de captura. |
| `search()` | Prova una coincidència en una cadena. Retorna l'índex de la coincidència, o `-1` si la recerca falla. |
| `replace()` | Executa una recerca per una coincidència en una cadena i reemplaça la subcadena coincident amb una subcadena de reemplaçament. |
| `replaceAll()` | Executa una recerca de totes les coincidències en una cadena i reemplaça les subcadenes coincidents amb una subcadena de reemplaçament. |
| `split()` | Utilitza una expressió regular o una cadena fixa per dividir una cadena en un arranjament de subcadenes. |

```
var expressio = /d(b+)d/g;
var matriu = expressio.exec('cdbbdbsbz');

var expressio = new RegExp('d(b+)d', 'g');
var matriu = expressio.exec(‘cdbbdbsbz');
```

+ Les expressions regulars tenen sis indicadors opcionals que permeten funcions com la recerca global i que no distingeixi entre majúscules i minúscules. Aquests indicadors es poden usar per separat o junts en qualsevol ordre i s'inclouen com a part de l'expressió regular.

Indicadors d'expressió regular

| bandera | Descripció | propietat corresponent |
|---|---|---|
| `g` | Cerca global. | `RegExp.prototype.global` |
| `i` | Cerca que no distingeix entre majúscules i minúscules. | `RegExp.prototype.ignoreCase` |
| `m` | Cerca multi-línia. | `RegExp.prototype.multiline` |
| `s` | Permet que l' `.` coincideixi amb caràcters de nova línia. | `RegExp.prototype.dotAll` |
| `u` | "Unicode"; tractar un patró com una seqüència de punts de codi Unicode. | `RegExp.prototype.unicode` |
| `y` | Realitza una recerca "enganxosa" que coincideixi a partir de la posició actual a la cadena de destinació. Consulta `sticky`. | `RegExp.prototype.sticky` |

```
var re = /\w+\s/g;
var str = 'fee fi fo fum';
var matriu = str.match(re);
console.log(matriu);

// ["fee ", "fi ", "fo "]
```

# BIBLIOGRAFIA



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