```
// Regular expressions are patterns
// used to match portions of strings
```

/ / **regular expression**

```
// All the code here checks itself
assert(/hello/.test('hello world'));
   // using Edward Hieatt's JsUnit
// from www.jsunit.net

// The test method says whether there's a match anywhere.
assert(!/i/.test("courage"));   // there is no "i" in courage
assert(/our/.test("courage"));   // there is "our" in courage

// The search method says how many characters precede.
assert("courage".search(/our/) == 1); // 1 letter before "our"
assert("courage".search(/i/)  == -1); // -1 means no match

// The replace method changes a matched substring
assert("recieve".replace(/ie/,"ei") == "receive");
```

character matching one-for-one:

A ... Z a ... z 0 ... 9 \

alphanumeric **punc-tuation**

\u0000 ... \uFFFF

Unicode hexadecimal

\x00 ... \xFF

ASCII hexadecimal

\0 ... \7
\00 ... \77
\000 ... \377

ASCII Octal

\cA ... \cZ

control characters

\0	[\b]	\t	\v	\f	\r	\n
NUL	back space	TAB	VT	FF	CR	new line
\x00	\x08	\x09	\x0B	\x0C	\x0D	\x0A

Character Classes (sets of matchables)

\d **digit** \D **non-digit**

```
// /\d/ matches any decimal digit
assert(/\d/.test('9'));
// same as /[0123456789]/ or /[0-9]/
assert(/[0123456789]/.test('9'));
assert(/[0-9]/.test('9'));
```

\s **space** \S **non-space**

```
// /\s/ matches space, tab, terminators
assert(/\S\S\s\S\S/.test("to be"));
// any invisible "white space" character
assert(/\s/.test(" "));
assert(/[ \t\n\u000B\f\r]/.test(' '));
```

\w **word char** \W **non-word**

```
// /\w/ is a letter, number, underscore
assert(/\w\w\W\w\w\w\W\w\w/
        .test("21-May/02"));
assert(/\w/.test('X'));
assert(/[0-9A-Za-z_]/.test('X'));
```

[ ] **one of**

```
var vowel=/[aeiouy]/;
assertEquals(2,'story'.search(vowel));
assert(!vowel.test('mfgr'));
```

[^] **one not of**

```
var nonvowel=/[^aeiouy]/;
assert(nonvowel.test('our'));
assert(!nonvowel.test('eye'));
```

- **range**

```
assert(/[a-z]/.test('Story'));
assert('$8ea'.replace(/[0-9]/,'X') == '$Xea');
assert(/[^a-zA-Z]/.test('Yes?'));
```

characters inside class [brackets]

A ... Z a ... z 0 ... 9 \

alphanumeric **punc-tuation**

\x0000 ... \xFFFF

Unicode hexadecimal

\x00 ... \xFF

ASCII hexadecimal

\0 ... \7
\00 ... \77
\000 ... \377

ASCII Octal

\cA ... \cZ

control characters

\0	\b	\t	\v	\f	\r	\n
NUL	back space	TAB	VT	FF	CR	new line
\x00	\x08	\x09	\x0B	\x0C	\x0D	\x0A

\d	\D	\s	\S	\w	\W
digit	non-digit	space	non-space	word char	non-word

```
// use digit, word or space in or outside class [ brackets ]
assert(/^[\d\s\(\)\-\+\/]*$/.test('(+20) 2 0900 0700'));
assert(/^[\d\s\(\)\-\+\/]*$/.test('714/921-5424'));
assert(!/^[\d\s\(\)\-\+\/]*$/.test('1-866-4MORTAL'));
// Slashed and Confused:
Backspace: /[\b]/ or '\b'  Not: /\b/ (word boundary)
Vertical tab: /\v/ or /[\v]/  Not: '\v' (same as 'v')
```