**定义和使用**

var patt1 = new RegExp("hello");

var patt2 = /world/ ;

**test方法**

test() 方法检索字符串中的指定值。返回值是 true 或 false。

var pat = /my/;

var str = "this is my code...";

console.log(pat.test(str)); // true

**exec方法**

exec() 方法检索字符串中的指定值。返回值是被找到的值。如果没有发现匹配，则返回 null。

var pat = /hello/;

console.log(pat.exec("oh hello world")); //返还hello

**正则表达式类型**

/pattern/attributes

参数 attributes 是一个可选的字符串，常用属性 "g"、"i" ，分别用于指定全局匹配、区分大小写的匹配。

//不区分大小写

var str = "Visit Hunger";

var patt1 = /hunger/i;

console.log(str.match(patt1)); //全局匹配

var str="hello hunger valley! I am hunger"; var patt1=/hunger/g; console.log(str.match(patt1)); //不区分大小写，全局匹配 var str="hello Hunger valley! I am hunger"; var patt1=/hunger/gi; console.log(str.match(patt1));

**字符串正则**

**1. search**

字符串查找

var str="Visit W3School!";

console.log(str.search(/w3school/)); //-1 console.log(str.serach(/w3school/i)); // 6

**2. match**

字符串匹配

var str="1 plus 2 equal 33";

console.log(str.match(/\d+/)); //[1]

console.log(str.match(/\d+/g)); //[1,2,33]

**3. replace**

字符串替换

var str="Hello JI! oh I am hunger"

console.log(str.replace(/Hunger/, "valley")); console.log(str.replace(/hunger/ig, "hunger"));

**4. split**

字符串分割

var str = "Hello Hunger , oh I am Hunger";

str.split("");

str.split(/\s+/);

**正则写法**

* [abc] 查找方括号之间的任何字符。

var str="Is this all there is?";

var patt1=/[a-h]/g;

console.log(str.match(patt1));

* [^abc] 查找任何不在方括号之间的字符。

var str="hello jikexueyuan!";

var patt1=/[^jike]/g;

console.log(str.match(patt1));

* [0-9] 查找任何从 0 至 9 的数字。
* [a-z] 查找任何从小写 a 到小写 z 的字符。
* [A-Z] 查找任何从大写 A 到大写 Z 的字符。
* [A-z] 查找任何从大写 A 到小写 z 的字符。
* [adgk] 查找给定集合内的任何字符。
* [^adgk] 查找给定集合外的任何字符。
* red|blue|green 查找任何指定的选项。

var str="hello hunger! How are you?";

var patt1=/hello|you/g; c

onsole.log(str.match(patt1));

* . 查找单个字符，除了换行和行结束符。

var str="That's hot!";

var patt1=/h.t/g;

document.write(str.match(patt1));

* \w 查找单词字符(字母、数字、下划线)。

var str="Give 100%!";

var patt1=/\w/g;

document.write(str.match(patt1));

* \W 查找非单词字符。

var str="Give 100%!"; var patt1=/\W/g; document.write(str.match(patt1));

* \d 查找数字。

var str="Give 100%!";

var patt1=/\d/g;

document.write(str.match(patt1));

* \D 查找非数字字符。

var str="Give 100%!"; var patt1=/\D/g; document.write(str.match(patt1));

* \s 查找空白字符(空格、tab、换行、回车)。

var str="Is this all there is?";

var patt1=/\s/g; document.write(str.match(patt1));

* \S 查找非空白字符。

var str="Is this all there is?"; var patt1=/\S/g; document.write(str.match(patt1));

* \b 匹配单词边界。

/\bm/ 匹配 "moon" 中的 'm'；

/oo\b/ 不匹配 "moon" 中的 'oo'，因为 'oo' 后面的 'n' 是一个单词字符；

/oon\b/ 匹配 "moon" 中的 'oon'，因为 'oon' 位于字符串的末端，后面没有单词字符； var str="Hello jikexueyuan";

var patt1=/\bjikexueyuan/g;

document.write(str.match(patt1));

* \B 匹配非单词边界。
* \n 查找换行符。

var str="Hello Hunger.\n be a FE.";

var patt1=/\n/g; document.write(str.search(patt1));

* n+ 匹配任何包含至少一个 n 的字符串。

var str="Hello HHunger! Hello World!";

var patt1=/H+/g;

document.write(str.match(patt1));

var str="Hello Hunger! Hello World!";

var patt1=/\w+/g; document.write(str.match(patt1));

* n\* 匹配任何包含零个或多个 n 的字符串。

var str="Hellooo Hunger! Hello World!"; var patt1=/lo\*/g; document.write(str.match(patt1))

* n? 匹配任何包含零个或一个 n 的字符串。

var str="1, 100 or 1000?"; var patt1=/10?/g; document.write(str.match(patt1));

* n{X} 匹配包含 X 个 n 的序列的字符串。

var str="100, 1000 or 10000?"; var patt1=/\d{4}/g; document.write(str.match(patt1));

* n{X,Y} 匹配包含 X 或 Y 个 n 的序列的字符串。

var str="100, 1000 or 10000?"; var patt1=/\d{3,4}/g; document.write(str.match(patt1));

* n{X,} 匹配包含至少 X 个 n 的序列的字符串。

var str="100, 1000 or 10000?"; var patt1=/\d{3,}/g; document.write(str.match(patt1));

* n$ 匹配任何结尾为 n 的字符串。

var str="Is this his"; var patt1=/is$/g; document.write(str.match(patt1));

* ^n 匹配任何开头为 n 的字符串。

var str="Is this his"; var patt1=/^Is/g; document.write(str.match(patt1));

**常见正则**

* 汉字： [\u4e00-\u9fa5]
* 手机号： 1[0-9]{10}
* 邮箱： (\S)+[@]{1}(\S)+[.]{1}(\w)+