



JAVASCRIPT CHEAT SHEET

- ▶ Programming language
- ▶ Strict syntax requirements
- ▶ Interpreted; object-oriented; dynamic typing
- ▶ Access web content (elements)
- ▶ Modify web content
- ▶ React to events
- ▶ Retrieve data without page reload (AJAX)

PRIMITIVE DATA TYPES

Type	Example(s)	Falsey Value(s)
number	1, 3, 5.6, -284	0, -0, NaN
string	"hello", "Mr. F"	""
boolean	true, false	false
null	null	null
undefined	undefined	undefined
symbol (ES6)	Symbol("first")	none

SOME TYPE CONVERSIONS

<code>parseFloat("0.05")</code>	Convert string to floating point number
<code>parseInt("1000", 10)</code>	Convert string to integer number in base 10
<code>89.toString()</code>	Convert number to string

OBJECTS

- ▶ used to store related data and methods
- ▶ key-value pairs
- ▶ syntax: { key: value, key2: value2 }
- ▶ dot notation: `obj.key = newValue;`
- ▶ bracket notation: `obj['key'] = newValue;`

```
pet = {
  name: 'Theo',
  bark: function(){
    console.log('woof!');
  },
  age: 3
}
```

ARRAYS

- ▶ used to store sequential data
- ▶ integer keys (indices) starting from 0
- ▶ syntax: [el0, el1, el2]
- ▶ bracket notation for indices: `arr[0] = 7;`

```
coinFlips = [0, 1, 1, 1, 0];
```

PROGRAMMING CONCEPTS

variable	<code>var thing = 5;</code>	Store some data and give it a name for reuse.
conditional	<pre>if (size == 'large'){ price = 4.99; } else if (size == 'medium'){ price = 2.00; } else { price = 1.29; }</pre>	Decide which line of code to run next based on current conditions.
loop	<pre>var j=10; while (j>=0;){ console.log(j); j=j-1; }</pre>	Repeat an action while a condition is true.
- initialization - continue condition - update	<pre>for (var k=10; k>=0; k=k-1){ console.log(k); }</pre>	Repeat an action for each value in a sequence.
function	<pre>function add (num1, num2){ return num1 + num2; }</pre> <pre>add(6, 9);</pre>	Function definition: store some behavior and give it a name for reuse. Function call: run the code now on specific values

BOOLEAN COMPARISON OPERATORS

<code>==</code>	Loose equal to	<code>></code>	Greater than
<code>===</code>	Strict equal to	<code><</code>	Less than
<code>!=</code>	Not equal to	<code>>=</code>	Greater than or equal to
<code>!==</code>	Strict not equal to	<code><=</code>	Less than or equal to

BOOLEAN LOGICAL OPERATORS

<code>&&</code>	and
<code> </code>	or
<code>!</code>	not