Learning Javascript Cheat Sheet

by kelt via cheatography.com/27757/cs/8119/

Comments

```
// single line comment
// great place to store your thoughts
/*
    this is a multi-line comment
    everything in here is ignored
*/
```

Conditions

```
if (person.favoriteColor === 'red')
{
    // give person a rose
}
else if (person.favoriteColor === 'blue') {
    // give person a violet
}
else {
    // tell person a poem
}
```

Loops

```
var year = 2016;
while (year < 2020)
{
   console.log("Party like it's ", year);
   // shorthand for: year = year + 1;
   year++;
}
for (var year = 2016; year < 2020; year++)
{
   // no need to increment year in here
   // because it is done in the for loop
   console.log('...', year);
}
```

Scope

```
// Scopes are controlled by brackets { }
// an outer scope cannot use variables
// declared inside inner scopes
// but an inner scope can use variables
// declared in the outer scope
// outside scope
var x = 1;
var y = 2;
var z = 3;
var result;
function testingScope ( z )
{
   // new variable inside function scope
   var w = 4;
   // y is scoped to this function
   // doesn't affect outer y
   var y = x;

   // z is a function parameter
   // so it's scope is local to function
   z = z + w;
   // updating x in outer scope
   x = 2;
  return z; // returns 8
}
result = testingScope(4);
/*
    w is undefined in outer scope
    x changed to 2
    y is still 2
    z still 3
    result is 8
```

Scope (cont)

```
*/
```

Keep Learning More

Use cheatsheets & docs

DOM cheat - https://christianheilmann.com/stuff/JavaScript-DOM-Cheatsheet.pdf

jQuery cheat - https://oscarotero.com/jquery/

jQuery docs - http://api.jquery.com/event.pagex/

javascript - https://www.cheatography.com/davechild/cheat-sheets/javascript/

quickly code - http://www.quicklycode.com/tag/javascript

Good Javascript books

Eloquent Javascript

Javascript the definitive guide

Variables

```
var answer    = 42;      // number
var duckSays  = 'quack'; // strings ' & "
var isAwesome = true;    // boolean
var emptiness = null;    // null
var catNames  = [        // array of strings
   'princess',
   'fizzy',
   'zoro'
];
```

Variables (cont)

```
var cat       = {         // a singl
e object
   name: catName[2],      // zoro
   age:  10
};
```

answer + 1;
// 43

duckSays + "quack";
// quack quack

catNames[2];
// zoro

cat.age = 11;
// sets cat age to 11

Intervals / Timeouts

```
function callMeShirley()
{
   console.log('Surely, we can
learn Javascript in 20
minutes?!');
}
function dontCallMeShirley()
{
   console.log('Don't call me
Shirely');
}
// runs every 1000 milliseconds (1
second)
var interval =
setInterval(callMeShirley, 1000);
// runs once after 5 seconds
var timeout =
setTimeout(dontCallMeShirley,
5000);
```

DOM / Document Object Model

The power to find, remove, replace, clone and create new html inside your web page
Nodes are different little bits and pieces of html
All nodes have a type (we can find it using, node.nodeType)

DOM / Document Object Model (cont)

```
ATTRIBUTE_NODE
CDATA_SECTION_NODE
COMMENT_NODE
DOCUMENT_NODE
DOCUMENT_FRAGMENT_NODE
ELEMENT_NODE
TEXT_NODE
```

... and a dozen others omitted ...
Elements are nodes that have a nodeType of `ELEMENT_NODE` such as

```
<div>
<p>
<a>
```

button element node

```
var button =
document.getElementById('specialBut
ton');
```

array of element nodes

```
var buttons =
document.getElementsByClassName('bu
tton');
```

new element node

```
var node =
document.createElement('div');
```

element node type

```
node.nodeType
```

cloned element node

```
var clone = node.cloneNode(true);
```

Functions

```
function addTogether(x, y)
{
   return x + y;
}
var z = addTogether(1, 2);
```

Functions (other ways)

```
var addEm = addTogether;
var add = function(x, y)
{
   return x + y;
}
var myObj = {
   add: function(x, y) {
      return x + y;
   }
}
myObj.addEm = addTogether;
```

All these functions return 2

addTogether(1, 1);
addEm(1, 1);
add(1, 1);
myObj.add(1, 1);
myObj.addEm(1, 1);

Classes / reusable objects

```
// constructor called for "new
Person"
function Person(name)
{
  this.name = name;
}
// all person's that are created
will have
// this function available
Person.prototype.changeName =
function(newName)
{
  this.name = newName;
}
var person = new Person("Bob");
person.changeName('Sponge');
```

Events

```
<button id="clickMe">Surely, You'll
Click Me</button>
var button =
document.getElementById('clickMe');
button.onclick = callMeShirley;
```

Often events are a better alternative than *intervals*. Events are fired only when triggered. *Intervals* happen regardless. In the above example, `callMeShirley` will only be triggered when the button is clicked.

Here is a list of events

jQuery

```
// get mouse position without
jQuery
document.onmousemove =
handleMouseMove;
function handleMouseMove(event)
{
  var dot, eventDoc, doc, body,
pageX, pageY;
  event = event || window.event;
  if (event.pageX == null &&
event.clientX != null)
  {
      eventDoc = (event.target &&
event.target.ownerDocument) ||
document;
      doc =
eventDoc.documentElement;
      body = eventDoc.body;
      event.pageX = event.clientX +
         (doc && doc.scrollLeft ||
body && body.scrollLeft || 0) -
         (doc && doc.clientLeft ||
body && body.clientLeft || 0);
      event.pageY = event.clientY +
         (doc && doc.scrollTop ||
body && body.scrollTop || 0) -
```

jQuery (cont)

```
         (doc && doc.clientTop ||
body && body.clientTop || 0 );
  }
}
// get mouse position with jQuery
$('body').on('mousemove',
function(event)
{
   var mousePosition = { x:
event.clientX, y: event.clientY };
});
```

jQuery is a library that can help us when we write javascript for the browser.

As seen in the example above, the jQuery library solves a problem for us. It abstracts away subtle browser inconsistencies for us.

Use jQuery whenever you can to make your code easier
to read, understand and maintain. There are also thousands of jQuery plugins too and you are likely to use some to enhance your website.