GWG - Javascript -

```
var name = "Ginny";
name[0] = G      or   name.charAt(0) = G
```

\\ backslash
\" \' " or '
\n newline
\t tab

null - value of null
undefined - does not have a value

```
if ( ) {          if () {
} else {          } else if {
}                 } else {
                  }
```

&& || !
and or not

AND			OR		
T	T	T	T	T	T
T	F	F	T	F	T
F	T	F	F	T	T
F	F	F	F	F	F

"" = F "any" = T
null = F 42 = T
undefined = F {} = T
0 = F 1 = T
NaN = F [] = T

TERNARY OPERATOR
conditional ? (if true) : (if false)

```
var color = yes ? "green" : "red";
var category = eatsPlants ? (eatsAnimals ? "omnivore" : "herbivore") :
                            (eatsAnimals ? "carnivore" : undefined);
```

GWG - JavaScript (cont.) -

```
switch (option) {
  case 1: /* code */ ;
  case 2: /* code */ ;
}
  break;
```

will fall through & execute all cases

```
while (x <= 10,000) {
      /* do something */
}
```

1. start → var x = 1;
2. stop → while (x < 10)
3. next → x = x + 1

```
for (var i = 0; i < 6; i = i + 1) {
      /* do something */
}
```

(1. 2. 3.)

```
for (start; stop; step) {
    // do this
}
```

```
nested - for (var x = 0; x < 3; x = x + 1) {
           for (var y = 0; y < 2; y = y + 1) {
             console.log(x + ", " + y);
}}
```

x	xcond	y	ycond.	
0	T	0	T	(0, 0)
0	T	1	T	(0, 1)
0	T	2	F	X
1	T	0	T	(1, 0)
1	T	1	T	(1, 1)
1	T	2	F	X
2	T	0	T	(2, 0)
2	T	1	T	(2, 1)
2	T	2	F	

increment & decrement operators:

x++ x-- x+=3 x-=3 x*=3 x/=3

GWG - Javascript (cont.) -

```
function functionName (parameter) {
    // do something
    return ;
}
```

parameter - variable name, appears in function declaration
argument - value passed in (as parameter) when function is called

HOISTING - Javascript hoists function & var declarations to the top of the current scope.
- variable assignments are not hoisted

function Expression - function stored in variable *not hoisted
* does not need a name

```
var catSays = function(max) {
    // code here
};
```

callback = function passed into another function

```
var catSays = function(max) {
  var catMessage = "";
  for (var i=0; i<max; i++) {
    catMessage += "meow";
  } return catMessage;
};
```

```
function helloCat(callback) {
    return "Hello" +
        callback(3);
}

helloCat(catSays);
```

GWG - JavaScript (cont) -

closure: anonymous in line function expressions

```
function movies (messageFunction, name) {
    messageFunction (name);
}
movies( function displayFavorite (movieName) {
   console.log("My favorite movie is " + movieName);
}, "Finding Nemo");
```

```
//anonymous function
var doThis = function(y) {
   return y+1;
};
```

```
// named function expression
var doThis = function addOne(y) {
   return y+1;
};
```

~~do Some~~ → doThis(5);

array.length (string.length)
array.push() → adds elements to end of array
array.pop() → removes elements from end of array
array.splice(startIndex, numberElementsto remove, elements you want to add)
array.reverse()
array.sort()
array.shift() → removes 1st element array
array.join("")

```
myArray.forEach(myFunction);
function &myFunction(element, index, array) {}
   donuts.forEach(function(donut) {
      donut += "hole";
   });
```

GWG - Javascript (cont)

```
var newArray = myArray.map(function(element) {
                    element = element + 100;
                    return element;
                });
```

arrays in arrays:

```
for (var r=0; r<grid.length; r++) {        *r=row
  for (var c=0; c<grid[r].length, c++) {   *c=col
    console.log(grid[r][c]);
} }
var grid = [[1, 2, 3, 4, 5],
            [6, 7, 8, 9, 10],
            [11, 12, 13, 14, 15],
            [16, 17, 18, 19, 20]];
```

object- `var umbrella = {};`

```
var umbrella = {
      color: "pink";
      isOpen: false
};
```

```
open: function() {
  if (umbrella.isOpen === true) {
    return "already open!";
  } else {
    umbrella.isOpen = true;
    returned "opened";
} }
```

```
umbrella.isOpen;   dot
umbrella["isOpen"]; bracket
umbrella.open();
```

*don't use numbers as first character of property
*no spaces or hyphens