Basics

On page script:

```
<script type="t ext /ja vas cri pt"> ... </s cri -
pt>
```

Include External script:

```
<script src="fi len ame.js " ></ scr ipt>
```

Deter

Use deter so that the JavaScript doesn't execute until after the DOM is loaded

```
<script src="sc rip t.j s" defer> </s cri pt>
```

Comments:

```
/* Multi line
      comment */
// One line
```

Outputs:

```
consol e.l og(a); // write to the browser console
docume nt.w ri te(a); // write to the HTML
alert(a); // output in an alert box
confir m("R eal ly? "); // yes/no dialog, returns
true/false depending on user click
prompt ("Your age?","0 "); // input dialog. Second
argument is the initial value
```

Data Types

```
var age = 18; // number
var name = " Jan e"; // string
var name = {first :"Ja ne", last:"D oe"}; // object
var truth = false; // boolean
var sheets = ["HT ML", " CSS " ,"JS "]; // array
var a; typeof a; // undefined
var a = null; // value null
Symbol type???
```

Objects:

```
var student = { // object name
        fir stN ame :"Ja ne", // list of properties
and values
```

Data Types (cont)

```
>    lastName:"Doe",
   age:18,
   height:170,
   fullName : function() { // object function
     return this.firstName + " " + this.lastName;
   }
};
student.age = 19; // setting value (add a property)
student[age]++; // incrementing
delete student.age; // delete a property
name = student.fullName(); // call object function
iterating:
for (key in object){
object[key];
}
```

Data Types evaluation

```
'' == '0' // false
'' == 0 // true
0 == '0' // true
NaN == NaN // false
[''] == '' // true
false == undefined // false
false == null // false
null == undefined // true
'' === '0' // false
'' === 0 // false
0 === '0' // false
NaN == NaN // still weirdly false
[''] === '' // false
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Data Types evaluation (cont)

> false === undefined // false
false === null // false
null === undefined // false

Dom Elements

Access an HTML element

```
docume nt.q ue ryS ele cto r('css select or');
docume nt.q ue ryS ele cto rAl l('css select or');
docume nt.g et Ele men tBy Id( " ele men -
tID " ).i nne rHTML = " Hello World! ";
example:
let element = docume nt.q ue ryS ele cto r(' #bu -
tton');
let allEle ments = docume nt.q ue ryS ele cto rAl -
l('css select or');
let elems = docume nt.q ue ryS ele cto rAl -
l('.quote, .comme nt'); // that have a
" quo te" class AND all elements that have a " -
com men t" class.
elemen t.s tyl e.top = '20px';
elemen t.s tyle.
```

Adding event listeners

```
Use dom function:
addEve ntL ist ene r(event name, function name);
```

Access attributes

```
you can't access the class attribute via
object.class
const image = docume nt.q ue ryS ele cto r(' img');
image.src ='somt hin g.jpg';
but you can't access the class attribute via
object.class
```

Access class attribute

```
classL ist.add and classL ist.re move:
const image = docume nt.q ue ryS ele cto r(' img');
image.c la ssL ist.ad d(' act ive');
image.c la ssL ist.re mov e(' hid den');
innerHTML: The raw HTML between the starting and
ending tags of an element, as a string
```

Dom Elements (cont)

> textContent: The text content of a node and its descendants. (This property is inherited from Node)

Create and remove elements

```
element = document.createElement(tag string)
element = document.createElement('h1')
containerElement.appendChild(element);
element.remove();
```

Style

Every element has a style attribute which has higher precedence than any css file.

```
element.style.font = 'something'
```

To undo a style set via the style attribute, you can set it to the empty string, the element will be styled according to any rules in the CSS file

```
element.style.font = ''
```

Conditionals

if else

```
if ((age >= 14) && (age < 19)) { // logical
condition
        status = " Eli gib le." ; // executed if
condition is true
} else { // else block is optional
        status = "Not eligib le." ; // executed if
condition is false
}
```

Switch statement:

```
switch (new Date().ge tDay()) { // input is
current day
       case 6: // if (day == 6)
               text = " Sat urd ay";
                break;
       case 0: // if (day == 0)
       case 1: // if (day == 1)
               text = " Sun day ";
                break;
```


Sponsored by ApolloPad.com
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Conditionals (cont)

```
>     default: // else...
       text = "Whatever";
}
```

? statements:

```
a = (condition) ? (cond_true_result) : (cond_false_result);
```

Numbers and Math

```
var pi = 3.141;
pi.toF ixe d(0); // returns 3
pi.toF ixe d(2); // returns 3.14 - for working with
money
pi.toP rec isi on(2) // returns 3.1
pi.val ueOf(); // returns number
Number (true); // converts to number
Number(new Date()) // number of millis econds
since 1970
parseI nt( "3 months "); // returns the first
number: 3
parseF loa t("3.5 days"); // returns 3.5
Number.MA X_VALUE // largest possible JS number
Number.MI N_VALUE // smallest possible JS number
Number.NE GAT IVE _IN FIN ITY// -Infinity
Number.PO SIT IVE _IN FIN ITY// Infinity
```

Math

```
var pi = Math.PI; // 3.1415 926 535 89793
Math.r oun d(4.4); // = 4 - rounded
Math.r oun d(4.5); // = 5
Math.p ow( 2,8); // = 256 - 2 to the power of 8
Math.s qrt (49); // = 7 - square root
Math.a bs( -3.14); // = 3.14 - absolute, positive
value
Math.c eil (3.14); // = 4 - rounded up
Math.f loo r(3.99); // = 3 - rounded down
Math.s in(0); // = 0 - sine
```

Numbers and Math (cont)

```
> Math.cos(Math.PI); // OTHERS: tan,atan,asin,acos,
Math.min(0, 3, -2, 2); // = -2 - the lowest value
Math.max(0, 3, -2, 2); // = 3 - the highest value
Math.log(1); // = 0 natural logarithm
Math.exp(1); // = 2.7182pow(E,x)
Math.random(); // random number between 0 and 1
Math.floor(Math.random() * 5) + 1; // random integer, from 1 to 5
```

Constants:

E, PI, SQRT2, SQRT1_2, LN2, LN10, LOG2E, Log10E

JSON

```
var str = '{"names":[' + // crate JSON object
'{"f irs t":"H aku na", " las tN": " Mat ata " },'
+
'{"f irs t":"J ane " ,"la stN " :"Do e" },' +
'{"f irs t":"A ir", " las t":"J ord an" }]}';
obj = JSON.p ars e(str); // parse
docume nt.w ri te( obj.na mes [1].fi rst); //
access
```

Send:

```
var myObj = { " nam e":"J ane ", " age ":18, " -
cit y":"C hic ago " }; // create object
var myJSON = JSON.s tri ngi fy( myObj); //
stringify
window.lo cation = " dem o.p hp? x=" + myJSON; //
send to php
```

Store:

```
myObj = { " nam e":"J ane ", " age ":18, " cit -
y":"C hic ago " };
myJSON = JSON.s tri ngi fy( myObj); // storing
data
localS tor age.se tIt em( " tes tJS ON", myJSON);
////
text = localS tor age.ge tIt em( " tes tJS ON"); //
retrieving data
obj = JSON.p ars e(t ext);
docume nt.w ri te( obj.name);
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Errors

```
try { // block of code to try
        und efi ned Fun cti on();
}
catch(err) { // block to handle errors
        con sol e.l og( err.me ssage);
}
```

Throw Error

```
throw "My error messag e"; // throw a text
```

Input Validation

```
var x = docume nt.g et Ele men tBy Id( " myn -
um").value; // get input value
try {
       if(x == " ") throw " emp ty"; // error
cases
        if( isN aN(x)) throw "not a number ";
       x = Number(x);
       if(x > 10) throw "too high";
}
catch(err) { // if there's an error
        doc ume nt.w ri te( " Input is " + err);
// output error
        con sol e.e rro r(err); // write the error
in console
}
finally {
        doc ume nt.w ri te( " </br />D one "); //
executed regardless of the try / catch result
}
```

Error name values

RangeError A number is "out of range"
Refere nce Error An illegal reference has occurred
Syntax Error A syntax error has occurred
TypeError A type error has occurred
URIError An encode URI() error has occurred

Cite

Got from http://htmlcheatsheet.com/js/ with customizations

Event types

Mouse

onclick, oncont ext menu, ondblc lick, onmous - edown, onmous eenter, onmous eleave, onmous emo - ve(only desktop), onmous eover, onmous eout, onmouseup

mouseE ven t.c lientX, mouseE ven t.c lientY: X|Y axis position relative to the left edge of the browser viewpoint

Touch

ontouc hca ncel, ontouc hend, ontouc hmo ve(only mobile), ontouc hstart

Pointe rEvent

(not for all browser can be used with polyfill) available with the following script:

```
<script src="ht tps :// cod e.j que ry.c om /pe -
p/0.4.1 /p ep.j s"> </s cri pt>
```

And we'll add need to add touch- act ion ="no ne" to the area where we want Pointe rEvents to be recognized (telling the browser that we do not want the default touch behavior for children of this element).

Works the same both on mobile and desktop

pointe rdown, pointerup, pointe rmo ve(works both on mobile and desktop), pointe rcancel

To listen to pointer events that occur when the pointer goes offscreen, call setPoi nte rCa pture on the target you want to keep tracking:

```
event.t ar get.se tPo int erC apt ure (ev ent.po -
int erId);
```

Keyword

onk eydown, onkeyp ress, onkeyup

keywor dEv ent.key: which key was pressed: Escape, ArrowR ight, ArrowLeft, ...

Frame

ona bort, onbefo reu nload, onerror, onhash - change, onload, onpage show, onpage hide, onresize, onscroll, onunload

Form

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Event types (cont)

> onblur, onchange, onfocus, onfocusin, onfocusout, oninput, oninvalid, onreset, onsearch, onselect, onsubmit

Drag

ondrag, ondragend, ondragenter, ondragleave, ondragover, ondrag-start, ondrop

Clipboard

oncopy, oncut, onpaste

Media

onabort, oncanplay, oncanplaythrough, ondurationchange, onended, onerror, onloadeddata, onloadedmetadata, onloadstart, onpause, onplay, onplaying, onprogress, onratechange, onseeked, onseeking, onstalled, onsuspend, ontimeupdate, onvolumechange, onwaiting

Animation

animationend, animationiteration, animationstart

Miscellaneous

transitionend, onmessage, onmousewheel, ononline, onoffline, onpopstate, onshow, onstorage, ontoggle, onwheel

Working with Events

Adding event listeners

Use dom function:

addEve ntL ist ene r(event name, function name);

Access caller element inside callback

event.t arget; //the element that was clicked / " dis patched the event" (might be a child of the target)

event.c ur ren tTa rget; // the element that the original event handler was attached to)

also 'this' refer to event in callback

Event propag ation

If a child and a parent element exist that have different callbacks to events, and inner element is clicked, both events are called. This is called event bubbling.

```
<div id=" out er">
    Click me!
```

Working with Events (cont)

```
>  <div id="inner">
   No, Click me!
 </div>
</div>
```

innerElem.addEventListener('click', innerCallback);

outerElem.addEventListener('click', oute/rCallback);

if inner element is called, outer and inner callbacks will be fired.

event.stopPropagation() method inside callback can prevent further propagation of event to outer callbacks.

the propagation can go in opposite direction (called event capturing) by setting additional parameter for addEventListener.

event.addEventListener('click', onClick, { capture: true});

prevent default

There's a default behavior for some events that can be prevented using preventDefault method.

```
function onCallback(event){
event.preventDefault();
}
```

Random number

Use Math.random: returns a random floating point number between [0, 1)

To get a random number in [0, maxNum ber):

Math.f loo r(M ath.ra ndom() * Max)

Data attributes

You can assign special data-* attributes to HTML elements to give associate additional data with the element

<tag data-y our -na me= 'your value"> </t ag>

to access in javasc ript:

tagEle men t.d ata set.Yo urName

Dash-s epa rated words turn to camel case and are returned as string

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Global functions

```
eval(); // executes a string as if it was script code
String (23); // return string from number
(23).t oSt ring(); // return string from number
Number ("23 "); // return number from string
decode URI (enc); // decode URI. Result: "my page.a sp"
encode URI (uri); // encode URI. Result: " my% - pag e.a sp"
decode URI Com pon ent (enc); // decode a URI component
encode URI Com pon ent (uri); // encode a URI component
isFini te(); // is variable a finite, legal number
isNaN(); // is variable an illegal number
parseF loat(); // returns floating point number of string
parseI nt(); // parses a string and returns an integer
```

Dates

```
Thu Mar 22 2018 11:40:54 GMT+0430 (+0430)
var d = new Date();
152170 2654616 milise conds passed since 1970
Number(d)
Date("2 017 -06 -23 "); // date declar ation
Date("2 017 "); // is set to Jan 01
Date("2 017 -06 -23 T12 :00 :00 -09 :45 "); // date - time YYYY-M M-D DTH H:M M:SSZ
Date("June 23 2017"); // long date format
Date("Jun 23 2017 07:45:00 GMT+0100 (Tokyo Time)"); // time zone
Get Dates:
Thu Mar 22 2018 11:40:54 GMT+0430 (+0430)
var d = new Date();
152170 2654616 milise conds passed since 1970
Number(d)
Date("2 017 -06 -23 "); // date declar ation
Date("2 017 "); // is set to Jan 01
```

Dates (cont)

> Date("2017-06-23T12:00:00-09:45"); // date - time YYYY-MM-D-DTHH:MM:SSZ
Date("June 23 2017"); // long date format
Date("Jun 23 2017 07:45:00 GMT+0100 (Tokyo Time)"); // time zone

Setting part of a date:

var d = new Date();
d.setDate(d.getDate() + 7); // adds a week to a date
setDate(); // day as a number (1-31)
setFullYear(); // year (optionally month and day)
setHours(); // hour (0-23)
setMilliseconds(); // milliseconds (0-999)
setMinutes(); // minutes (0-59)
setMonth(); // month (0-11)
setSeconds(); // seconds (0-59)
setTime(); // milliseconds since 1970)

Regular Expressions

Promises

```
An object used to manage asynchronous results.
has a then() method that lets you attach functions
to execute onSuccess and onError callbacks.
Allows you to build chains of asynch ronous
results.
```

Local Storage

Variables

```
var a; // Function scope variable
let al; // block scope variable
const ca; // block scope constant, cannot be
changed. However, it doesn't provide true const
correc tness, so you can still modify the
underlying object so:
const list = [1, 2, 3];
list.p ush(4); // OK
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Variables (cont)

> var b = "init"; // string
var c = "Hi" + " " + "Joe"; // = "Hi Joe"
var d = 1 + 2 + "3"; // = "33"
var e = [2,3,5,8]; // array
var f = false; // boolean
var g = /()/; // RegEx
var h = function(){}; // function object
const PI = 3.14; // constant
var a = 1, b = 2, c = a + b; // one line
let z = 'zzz'; // block scope local variable

Strict mode

"use strict"; // Use strict mode to write secure code
x = 1; // Throws an error because variable is not declared

Operators

Values

```
false, true // boolean
18, 3.14, 0b10011, 0xF6, NaN // number
" flo wer ", 'John' // string
undefined, null , Infinity // special
```

Operators

```
a = b + c - d; // addition, substr action
a = b * (c / d); // multip lic ation, division
x = 100 % 48; // modulo. 100 / 48 remainder = 4
a++; b--; // postfix increment and decrement
```

Bitwise operators

```
& AND 5 & 1 (0101 & 0001) 1 (1)
| OR 5 | 1 (0101 | 0001) 5 (101)
NOT 5 (~0101) 10 (1010)
XOR 5 1 (0101 ^ 0001) 4 (100)
```

Operators (cont)

> << left shift 5 << 1 (0101 << 1) 10 (1010)
>> right shift 5 >> 1 (0101 >> 1) 2 (10)
>>> zero fill right shift 5 >>> 1 (0101 >>> 1) 2 (10)

Asthmatics:

a * (b + c) // grouping
person.age // member
person[age] // member
!(a == b) // logical not
a != b // not equal
typeof a // type (number, object, function...)
x << 2 x >> 3 // minary shifting
a = b // assignment
a == b // equals
a != b // unequal
a === b // strict equal
a !== b // strict unequal
a < b a > b // less and greater than
a <= b a >= b // less or equal, greater or eq
a += b // a = a + b (works with - * %...)
a && b // logical and
a || b // logical or

Strings

```
var abc = "abcdefghijklmnopqrstuvwxyz";
var esc = 'I don\'t \n know'; // \n new line
var len = abc.le ngth; // string length
abc.in dex Of( " lmn o"); // find substring, -1 if
doesn't contain
abc.la stI nde xOf ("lm no"); // last occurance
abc.sl ice(3, 6); // cuts out " def ", negative
values count from behind
abc.re pla ce( " abc " ,"12 3"); // find and
replace, takes regular expres sions
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Strings (cont)

> abc.toUpperCase(); // convert to upper case
abc.toLowerCase(); // convert to lower case
abc.concat(" ", str2); // abc + " " + str2
abc.charAt(2); // character at index: "c"
abc[2]; // unsafe, abc[2] = "C" doesn't work
abc.charCodeAt(2); // character code at index: "c" -> 99
abc.split(","); // splitting a string on commas gives an array
abc.split(""); // splitting on characters
128.toString(16); // number to hex(16), octal (8) or binary (2)

Functions

Declaration

```
functions in JavaScript are " fir st- cla ss"
because they are treated like any other variab -
le/ object.
function hello( input) {
// body
}
hello = (input) => {
// body
}
// you can omit {} if there is only one statement:
// you can omit () if there is only one argument
hello = input => input + 1;
var hello = function hello( input) { // body }
```

this scope inside functions

```
functions:
this is will be dynami cally assigned to a
different value depending on how the function is
called (unless explicitly bound with bind)
- this in a constr uctor: refers to the new object
being
created
- this outside a constr uctor: refers to a
different value
```

Functions (cont)

> depending on how the function is called
  - In response to a DOM event, this is the element
  that the event handler was tied to
  - When called in a method, this is the object that
  the method is called from
- bind: sets the value of this for a function so it does
not change depending on the context
Arrow functions link:
Arrow functions capture the this value of the enclosing context.
Arrow function syntax uses “lexical scoping”. Lexical scoping is fancy way of saying it uses “this” from the surrounding code… the code that contains the code in question.
Arrow functions do not have their own this.
arrow functions cannot be used as generators (yeild does not work unless for further nested function)

```
this.test = "attached to the module";
var foo = {
  test: "attached to an object"
};
foo.method = function(name, cb){
  this[name] = cb;
};
foo.method("bar", () => {
  // not what you expected, maybe?
  console.log(this.test);
});
```

foo.bar(); // => "attached to an object" but it was expected to be madule!
As another example:

```
var obj = {
  i: 10,
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Functions (cont)

```
>   b: () => console.log(this.i, this),
  c: function() {
    console.log(this.i, this);
  }
}
obj.b(); // prints undefined, Window {...} (or the global object)
obj.c(); // prints 10, Object {...}
```

this inside object as function:

```
obj = { 'a': 12, f: function(){ console.log(this.a)}}; obj.f() // => 12
```

this inside object as arrow function:

```
obj = { 'a': 12, f: () => { console.log(this.a)}}; obj.f(); // => undefined.
```

or:

```
a = 1; obj = { 'a': 12, f: () => { console.log(this.a)}}; obj.f(); // => 1
```

functional programming:

- First-class functions (functions as objects)
- Closures
- Anonymous functions / lambdas / function literals
- Currying

Hoisted declaration

Some function declaration are hoisted. definition gets moved to the top of the scope in which it's defined. So this works:

```
hello()
function hello() { // ,,, }
```

These kind of declaration are not hoisted:

closures

A function declared within a function is also known as a closure. Inner functions are called closures

```
function outerFunc() {
 function innerFunc() {
```

Functions (cont)

```
>  // closures
 }
}
```

Scope:

Functions declared with function (or var) have function scope

Functions declared with const or let have block scope

Access to outer function variables

closures have access to variables declared in outer function and those variables does not go away! (remains and can be even changed!!). It's like a function pointer but local variables of outer function remained and are not destroyed.

```
f = function outerFunction(){
 let x = 1;
 return function(){
 x = x + 1;
 console.log(x);
 }
}
run = f();
run(); // 2
run(); // 3
run(); // 4
```

curryng

The idea of constructing a new function that is "partially instantiated" with arguments is called currying.

Constructing a new function that references part of the outer function's parameters is called currying.

for example when one function needs additional argument and we can't add another argument to it (it's not expected to have another one!), we can use currying. we can have a function that accepts that additional argument and have another function (a closure) which has expected arguments:

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Functions (cont)

> we want a function with 2 arguments to have another argument:
somthing.add(twoArgumentFunc)
function makeTwoArgument(anotherArg){
 return function(){
  // use anotherArg in here
  // It's like having 3 arguments!
 }
}
=> somthing.add(makeTwoARgument(anotherArg));

Anonymous function

When we define a function without an identifier, we call it an anonymous function

return function() {
 //this is anonymous function
}

Arrays

```
var dogs = ["Bulldog", "Beagle", "Labrador"];
var dogs = new Array( " Bul ldo g", " Bea gle ",
" Lab rad or"); // declar ation
alert( dog s[1]); // access value at index, first
item being [0]
dogs[0] = "Bull Terier "; // change the first item
for (var i = 0; i < dogs.l ength; i++) { //
parsing with array.l ength
        con sol e.l og( dog s[i]);
}
```

Methods:

```
dogs.t oSt ring(); // convert to string: results
" Bul ldo g,B eag le, Lab rad or"
dogs.j oin ('"); // join: " Bulldog Beagle *
Labrad or"
dogs.p op(); // remove last element
dogs.p ush ("Ch ihu ahu a"); // add new element to
the end
dogs[d ogs.le ngth] = " Chi hua hua "; // the same
as push
dogs.s hift(); // remove first element
```

Arrays (cont)

> dogs.unshift("Chihuahua"); // add new element to the beginning
delete dogs[0]; // change element to undefined (not recommended)
dogs.splice(2, 0, "Pug", "Boxer"); // add elements (where, how many to remove, element list)
var animals = dogs.concat(cats,birds); // join two arrays (dogs followed by cats and birds)
dogs.slice(1,4); // elements from [1] to [4-1]
dogs.sort(); // sort string alphabetically
dogs.reverse(); // sort string in descending order
x.sort(function(a, b){return a - b}); // numeric sort
x.sort(function(a, b){return b - a}); // numeric descending sort
highest = x[0]; // first item in sorted array is the lowest (or highest) value
x.sort(function(a, b){return 0.5 - Math.random()}); // random order sort
splice // Add/remove element at index
list.splice(startIndex, deleteCount, item1, item2, ...)
list.splice(3, 1) //Remove one element at index 3:
list.splice(2, 0, element); //Add element at index 2:
list.forEach(function): Executes the provided function once for each array element.
list.filter(function): Creates a new array with all elements that pass the test implemented by the provided function.
list.every(function) : Tests whether all elements in the array pass the test implemented by the provided function
others:
 concat, copyWithin, every, fill, filter, find, findIndex, forEach, indexOf, isArray, join, lastIndexOf, map, pop, push, reduce, reduce-Right, reverse, shift, slice, some, sort, splice, toString, unshift, valueOf

Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Loops

For Loops:

```
for (var i = 0; i < 10; i++) {
        doc ume nt.w rite(i + ": " + i*3 + " <br
/>");
}
var sum = 0;
for (var i = 0; i < a.length; i++) {
       sum + = a[i];
} // parsing an array
```

For each

```
for ( var i in someth ing){
    con sol e.l og( som eth ing [i]);
}
```

While Loops:

```
var i = 1; // initialize
while (i < 100) { // enters the cycle if statement
is true
       i *= 2; // increment to avoid infinite loop
        doc ume nt.w rite(i + ", "); // output
}
```

Do While Loops:

```
var i = 1; // initialize
do { // enters cycle at least once
       i *= 2; // increment to avoid infinite loop
        doc ume nt.w rite(i + ", "); // output
} while (i < 100) // repeats cycle if statement is
true at the end
```

Break:

```
for (var i = 0; i < 10; i++) {
       if (i == 5) { break; } // stops and exits
the cycle
        doc ume nt.w rite(i + ", "); // last output
number is 4
}
```

Loops (cont)

> Continue:

```
for (var i = 0; i < 10; i++) {
   if (i == 5) { continue; } // skips the rest of the cycle
   document.write(i + ", "); // skips 5
}
```

Classes

Definition

```
class ClassName{
    con str uct or(){ //it's optional
       // define public fields by setting this.f -
iel dName
       // in constr uctor or any method
        thi s.f iel dName = value;

   }

    met hod Name(){ //don't need to use function
keyword
        //all methods are public - no private
method yet
       // conven tion: _ before the name means
private
   }
    met hod Two(){
    //a lways refer to other methods or fields in
class with this
        thi s.m eth odN ame();

   }
}
```

Element as a class

Use class that gets Container element as input for constr uctor

Definition

```
class ClassName{
    con str uct or( con tai ner Ele ment){
        thi s.c ont ain erElem = contai ner Ele -
ment;

        //C reate element:
```


Sponsored by ApolloPad.com
Everyone has a novel in them. Finish Yours!
https://apollopad.com

Classes (cont)

```
>     this.image = document.createElement('img');
    this.image.src = 'image.jpg';
    this.image.addEventListener('click', this._clickElement)

  }

  _clickElement(event){
    // do some thing
  }
}
```

Binding this

This in here refers to the element that the event handler was attached to!

to make this always refer to the instance object for a method in the class we use bind

for above example:

first bind this in the constructor:

```
 constructor(containerElement){
 // always bind event listeners in the constructor
  this._clickElement = this._clickElement.bind(this);
 }
```

then use this as expected:

```
clickElement(event){
// now this refer to the class object not event element!
    this.image.src = 'image2.jpg';
}
```

another solution was to access the element with event.currentTarget

Instantiation

```
const x = new SomeClass();
const y = new SomeClass();
y.someMethod();
```

Custom events

```
You can listen to and dispatch (fire) Custom
Events.
const event = new Custom Eve nt( eve ntN ame -
String, option alP ara met erO bject);
elemen t.a ddE ven tLi ste ner (ev ent Nam eSt -
ring);
elemen t.d isp atc hEv ent (ev ent Nam eSt ring);
Custom Event can only be listened to /dispa tched
on HTML elements, and not to arbitrary class
instances.
```

Commun icating between element

```
You can add custom event on the document and
dispatch that event inside different element.
docume nt.a dd Eve ntL ist erner( ...
docume nt.d is pat chE ven t(...
to add additional inform ation about the event you
can pass parameter to the custom event:
onClic k(e vent) {
  const eventInfo = {
  but tonName = this.text;
 }
  doc ume nt.d is pat chE vent( new Custom Eve -
nt( 'bc lic ked', { detail: eventi nfo});
}
onCust omE ven (event) {
event.d et ail.bu tto nName; // can access info
}
```

Template literals

```
Template literals allow you to embed expressions
in
JavaScript strings
const port = 80;
consol e.log$erver is listening on port: ${port
});
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com

async/await

Fetch

```
The API to use to load external resources (text,
JSON, etc) in the browser (like XHR but easier to
use)
- It has only one function: fetch
- takes string path to the resource as argument
- return a promise
Fetch api:
function onSucc ess (re sponse) {
...
}
function onFail (re sponse) {
...
}
fetch( 'im age s.t xt' ).t hen (on Suc cess,
onFail);
or
const promise = fetch( 'im age s.t xt');
promis e.t hen (on Suc cess, onFail);
```


Sponsored by **ApolloPad.com**
Everyone has a novel in them. Finish Yours!
https://apollopad.com