Computernavn :_____

Opgaver:

1. Åben Visual Web Developer, vælg menuen **File** -> **New Web Site**.
Vælg template **Empty Web Site,** sæt location til **HTTP** og stien til **http://localhost/Javascript** og lad language stå til **C#.** Klik til sidst på **OK**.
I solution explorer (højre side panel) højre klikkes på "http://localhost/Javascript" her vælges nu **New Item,** templatesiden **HTML Page** vælges og navngives default.htm. Opret nu en webside med nedenstående udseende. Gem den ved at trykke på **CTRL-S**

JavaScript Links til opgaver opgave 1 opgave 2 .

Lav en mappe på webben med navnet **opgaver** (laves i solution explorer)**.** Opret i den mappe en web-side med titlen "Opgave 1". Siden skal indeholde scriptet fra præsentationen. Afprøv scriptet. Indsæt følgende kode i slutningen af script sektionen:

```
alert ('Resultatet blev  ' + c);
status='All done!';
```

Gem siden som opgave1.htm og indsæt et link i default.htm, der henviser til denne side. I de efterfølgende opgaver skal i selv huske at opdaterer default-siden med links til opgaverne.

2. Opret en side med navnet opgave2.htm, der ved hjælp af en "confirm" beder om bekræftelse på om brugeren er over 18år. Hvis brugeren er over 18 år udskrives en boks med besked om at brugeren har stemmeret. Brug en **if-else** struktur i denne opgave. Se eksemplet herunder

```
if  (testudtryk)
    { kode ved sandt testudtryk }
else
    { kode ved falsk testudtryk }
```

3. Modificer opgave 2 således at hvis brugeren ikke over 18, spørges om hvor gammel brugeren er. Det beregnes og udskrives hvor lang tid der er til brugeren opnår stemmeret.

4. Opret et array, der kan konvertere et terningeslag (værdi 1-6 indtastet i vha prompt) til teksten etter, toer … sekser. Resultatet skal vises i en alert-box.

5. Lav et script, der indlæser en radius vha. message bokse, og derefter udskriver den beregnede omkreds (document.write (2*Math.PI*radius)). Afprøv programmet. Modificer programmet således at brugeren spørges om han ønsker arealet beregnet (confirm), er det tilfældet beregnes og udskrives arealet (document.write (Math.pow(radius,2)*Math.PI)), ellers udskrives omkreds.

6. Opret en funktion der laver et sidehoved med opgavenavn, og en som laver en sidefod med redigeringsdato. Disse scripts skal være i en ekstern fil.

7. Lav en validering af input værdier i programmet fra opgave 5, således at kun radier i intervallet 5-10 accepteres (do …. while).

8. Opret et script, der dynamisk, det vil sige script genereret, udskriver to-tabellen i en HTML-tabel.

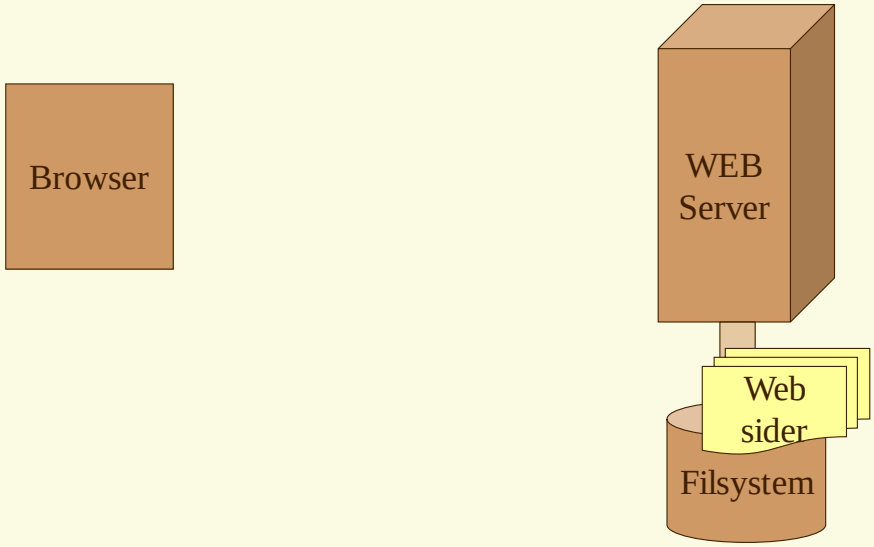
9. Opret formularvalidering af alle felter i den udleverede formular, kriterier for de enkelte felter aftales med instruktøren.

10. Opret en formular med inputfelt til radius, en checkboks til markering af om areal ønskes beregnet i stedet for omkreds samt en boks, der viser det beregnede areal. En checkboks har **true** som værdi i egenskaben **checked** når feltet er afkrydset. Beregningen skal være lavet som en funktion, der kaldes når der klikkes på en knap.

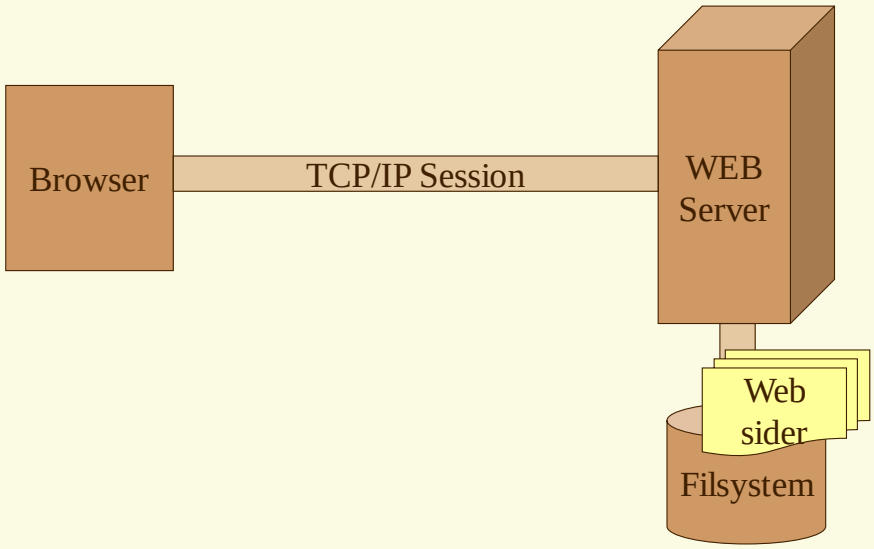
11. Modificer forrige opgave således at resultatet udskrives i en HTML-tag vha. dennes **innerHTML** egenskab.

12. Case (f.eks. menusystem) i samarbejde med instruktøren.

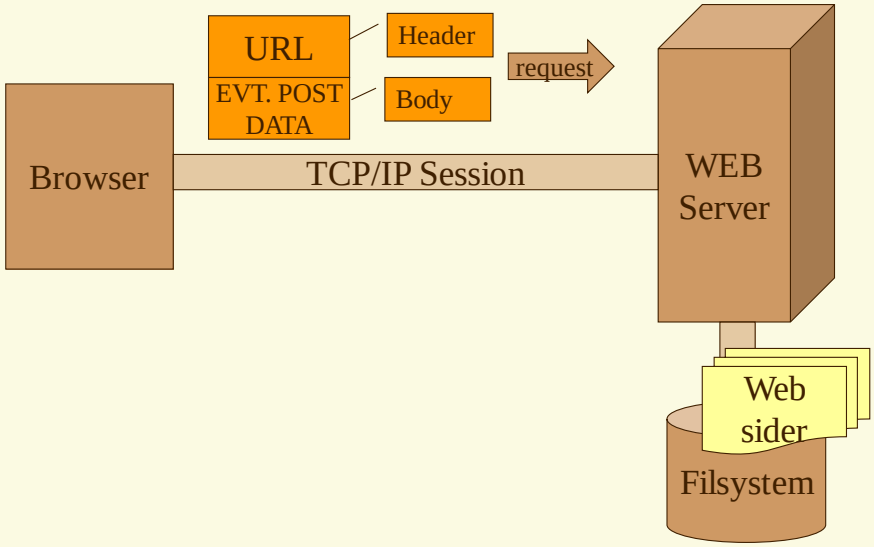
HTTP request - response



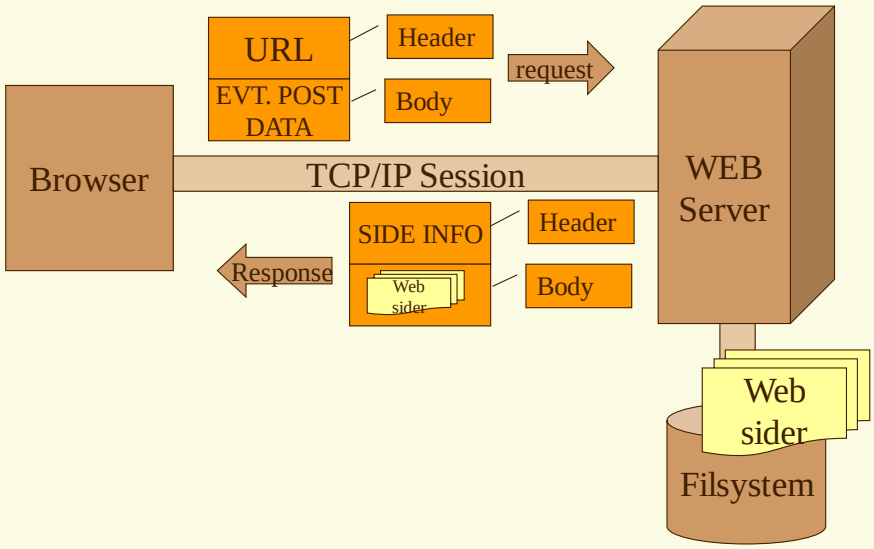
HTTP request - response



HTTP request - response



HTTP request - response



Introduktion til JavaScript

- Anvendelsesområder.
- Muligheder med JavaScript.
- Script-fortolkeren – sprogets motor.
- Er JavaScript et programmeringssprog ?
- Integration med websider.
- Objekter.
- JavaScript sproget . . . . . .

Anvendelsesområder

- Primært på HTML sider, som det der giver dynamik og intelligens på siden.
- Kan også anvendes på webservere, som det der muliggør dynamisk genereringen af HTML sider.
- Samt i Windows, som afløseren for batch-programmering.

Muligheder med JavaScript

- Er programmeringsprog for WEB-side udviklere.
- Kan dynamisk generere HTML-koder og indhold, samt aflæse selv samme indhold.
- Kan reagere på hændelser i browseren / på websiden.
- Kan validere data i web-formulare.
- Kan oprette/aflæse cookies i browseren.

Script-fortolkeren

- JavaScript er, som navnet jo også antyder, et fortolket programmeringssprog. Det betyder at alle kommandoerne køres af et separat program kaldet script fortolkeren. Denne fortolker afvikler JavaScript programmets linjer – linje for linje og er normalt indbygget i browseren.

Er JavaScript et programmeringssprog ?

- For "Professionelle" programmører er det kun sprog i stil og omfang som C++, der betragtes som "rigtige" sprog.
- JavaScript er et programmeringssprog, det er bare designet til at løse andre opgaver end typisk applikations udvikling.
- Og de "små" opgaver løser et scriptsprog bedst !

JavaScript og HTML

- Indsættes i HTML mellem Script tags

```
<h2>JavaScript eksempel</h2>
<script type="text/javascript">
        var a=5;    //Opret og tildel variablen a værdien 5
        var b=6;    //Opret og tildel variablen b værdien 6
        var c=a+b; //Opret c og tildel den summen af a og b

        document.write (c);  //udskriv resultat
</script>
```

```
<script language=javascript>
<!--
  …code  (”old school”)
//-->
</script>
```

```
<script type="text/javascript">
<![cdata[
… code (xhtml)
]]>
</script>
```

Objekter

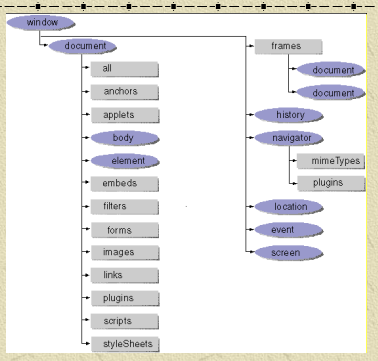
- I JavaScript er så godt som alle de elementer, sproget kan håndtere (dette inkluderer også variable) implementeret som objekter.
- Objekter har flg. karakteristika
  - Egenskaber //værdier objektet kan gemme
  - Metoder //funktioner objektet kan udføre
  - Events //signaler som objektet kan sende

Objekter

- JavaScript har ikke selv nogle kommandoer til styring af browseren. De skal leveres som objekter af den browser JavaScript afvikles i. Disse objekter kan JavaScript anvende til at udføre sit arbejde.
- Beskrivelsen af en browsers objekter benævnes en Objekt Model.
- Disse objekter kaldes for DOM objekter

Document Object Model (DOM)

Internet Explorer Objekt Model



EVENTS

- Hændelser på en HTML-side, håndteres af JavaScript funktioner.

```
<script type="text/javascript" >
function handler(){
        alert('Ska\' du Pille!');
}
</script>
<form name=frmTest>
<input type="button" name="ok" value="Rør mig"
        onmouseover="handler();">
</form>
```

Datatyper

- JavaScript tilbyder tre simple datatyper:
  - Tal (optræder som et Number objekt)
  - Tekst (optræder som et String objekt)
  - Sand/Falsk (optræder som et Boolean objekt)
- Samt mere komplekse datatyper:
  - Tabeller (optræder som et Array objekt)
  - Datoer (optræder som et Date objekt)
  - Objekter (objects)

Datatyper

- Eksempler på oprettelse af variable:
  - *var* kan udelades, men er ikke god "skik"
    - var i=5;
    - var f=12.25;
    - var s='Javascript';
    - var b=true;

Datakonvertering

- I JavaScript findes to indbyggede funktioner til konvertering af tekststrenge til numeriske variable:
  - var i = parseInt('234'); //sæt i=234
  - var f = parseFloat('12.50'); //sæt f = 12.5
- Operatoren **typeof** anvendes til at bestemme en variables type.
  - document.write(typeof(f)); //skriver number

Datatyper

- Eksempel med komplekse datatyper:
  - var nu = new Date(); (dato/tid lige nu)
  - var tomtabel = new Array();
  - var talnavne = new Array('nul','et','to');
  - var biler = ['Opel', 'BMW', 'Audi'];
- Arrays starter altid med indeks 0
- I JavaScript er der ikke umiddelbart mulighed for arrays i mere end dimension.

Variabel etik !!

- Variable skal ikke, men bør altid oprettes med en **var** sætning før ibrugtagning.
- Variable bør navngives så man antyder anvendelsen.
- **HUSK det nu** ! Der **er** forskel på store og små bogstaver. Også i objekt og funktions navne.

Data input/output fra JavaScript !

- Message bokse
  - alert ('besked til brugeren');
  - tal = prompt ('ledetekst', 'forslag');
  - svar = confirm('spørgsmål');
- Browserens document objekt
  - document.write ('<h1>Overskrift</h1>');
  - tal = document.forms['frm'].txtInput.value;

Program strukturer

- JavaScript anvender funktioner til at opdele programmet i logiske dele. Disse funktioner kan så kaldes efter behov. Funktioner kan modtage 0 eller flere **parametre**. Og har mulighed for at antage en værdi - en **retur værdi**. Funktioner bruges meget ofte til at definerer hvad et event skal afstedkomme, disse funktioner kaldes så *eventhandlere*.

Funktion

```
function dato(){
   var idag=new Date();
   document.write (idag);
}
document.write ('Det er i dag ');
dato();
document.write(' tillykke');
```

Funktion med parameter

```
function overskrift(linie){
    document.write ('<h1 align=center>');
    document.write (linie);
    document.write ('</h1>');
}
overskrift('Javascript eksempler');
overskrift('der forklarer!');
```

Funktion med returværdi

```
function TilCelcius(fahrenheit){
    return ( (fahrenheit-32) * 5/9) ;
}

var t=90;
document.write ('Hej temperaturen er '+TilCelcius (t)+'C');
document.write ('Hello the temperature is '+t+'F');
```

Programmets adgang til websiden

- Som det tidligere er nævnt, kan JavaScript programmet tilgå browseren og dens indhold gennem objekter. Den objekt model der er til rådighed i dag DOM, giver os mulighed for at kontrollere **alle** elementer i browseren.

Læsning af et formularfelt

```
<script type="text/javascript" >
function handler(){
    alert('Hej ' + document.forms['readname'].navn.value);
}
</script>

<form  name="readname">
Navn  <input type="text" name="navn">
          <input type="button" name="ok" value="OK"
                    onclick="handler()">
</form>
```

Operatorer

- JavaScript indeholder fire typer operatorer:
  - Beregning
    - talværdier og tekststrenge (+, -, *, / , %, ++, --)
  - Sammenligning
    - talværdier og tekststrenge (= =, >, >=, <=, <, !=)
  - Logiske
    - udsagn (sandt eller falsk) (&&, ||, !)
  - Tildeling
    - til variable (=, += …..)

Tekst sammenføjning.

- Til sammenføjning af tekst, og indsættelse af variable i tekst bruges + operatoren:
  - var navn ; var efternavn='Henriksen';
  - navn ='John' + ' ' + efternavn;

Programstrukturer

- if (udtryk) - else
- max = (a>b) ? a:b
- while (udtryk)
- do - while(udtryk)
- for (init; udtryk; update)
- for (var in object)
- switch (udtryk) case … default

if else

```
var a;
a=prompt('Indtast et tal', '5');

if (a > 100){
   alert ('Stort tal!');
}
else
{
    alert ('Lille tal!');
}
```

Betinget udtryk

```
//vend mønten ☺
var moent= new Array('plat','krone');
side=0;                        //plat;
side = (side==0) ? 1:0   //krone;
side = (side==0) ? 1:0   //plat;
side = (side==0) ? 1:0   //krone;
document.write (moent[side]);
```

while udtryk

```
var  b=0,  a = 12;

while (a  > 5){
   b +=12;
   a--;
}
alert('værdien af b ='+b); //hvad udskrives?
```

do while

```
var tal;

do{
    tal =prompt('Indtast værdi mellem 5 og 10')
}while(tal < 5 || tal >10);

alert('Tallet er ' + tal);
```

for

- For loopen gentager en kode del et bestemt antal gange.

```
var a, tekst='';

for (a=2;  a<=20;  a+=2){
        tekst = tekst + a + '\n';
}

alert('Her er 2 tabellen' + '\n'+ tekst);
```

for (var in obj)

- For in loopen anvendes til at gennemløbe et objekts egenskaber eller et array fra start til slut.
- Her er et eksempel hvor browserens navigator objekt "tappes" for informationer

```
var i;
 for(i in navigator)
 {
        document.write(i+' = '+navigator[i]+'<br/>');
 }
```

break –Så er det slut ☺

- Ønsker man at ændre forløbet i en af de tidligere nævnte løkker kan man anvende **break,** som afslutter eller **continue,** der genstarter løkken.

```
var i = 0;
while (i < 10) {
  i++;
  if (i==7)  continue;
  document.write(i + ".<br/>");
}
```

switch – vælg det rigtige ☺

```
var input;
input = prompt('Indtast valg: a,b eller c','a');
switch(input){
        case 'a': case 'A':
                alert('Du valgte a');
                break;
        case 'b': case 'B':
                alert('Du valgte b');
                break;
        case 'c': case 'C':
                alert('Du valgte c');
                break;
        default:
                alert('Du valgte forkert');
}
```

JavaScript Programmering

Eksempel 1:

Dette eksempel viser anvendelsen af **document objektets write metode,** samt **window objektets status egenskab**. Bemærk at der ved kald af en metode anvendes parenteser til at omkranse den tekst, som skal udskrives. Når en egenskab skal tildeles en værdi, som f.eks. status, bruges et lighedstegn på samme måde som når en variabel tildeles en værdi.

Tekst skal altid omkranses af apostroffer som i; 'Jeg er en glad lille cowboy'

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Javascript eksempler</title>
<script type="text/javascript" >

   //evt. scriptkode der afvikles før dokumentet vises,

</script>
</head>
<body>
<h1 align="center">Javascript eksempel</h1>
<script type="text/javascript" >

   //scriptkode der afvikles under
   //sideopbygningen indsættes her
   //window er default objektet og skal ikke altid skrives

   document.write('<h2 align=center>Hello javascript world</h2>');

   //lad variable i gå fra 1 til 5
   for(i=1; i<6; i++){

       //document.write('i er = ' + i + '<br>');
       document.write('<h'+i+' align=center>Javascript styrer!</h'+i+'>');

   }

   status='Hello world';

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 2:

Dette eksempel viser hvordan det er muligt at indsætte JavaScript kommandoer, som **eventhandler** for HTML tags**. window.navigate metoden** gør det man kunne forvente; lader browseren gå til en ny webside. Dette er dog også muligt ved hjælp af **location objektets href egenskab**; location.href='http://www.google.dk'

Bemærk den strukturerede måde **h1** taggen er opdelt på!

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Javascript eksempler</title>
</head>

<body>

<h1 onmouseover=" status='AV!'; "
    onmouseout="  status='Bliv dog ved'; "
>Prøv at gøre noget på mig ;-)</h1>

<h2 onclick=" window.navigate('http://www.google.dk'); ">"Link" til Google</h2>

</body>
</html>
```

JavaScript Programmering

Eksempel 3:

Dette eksempel viser hvordan det er muligt at få noget JavaScript kode til at køre ”forsinket”. I dette tilfælde køres funktionen skift() først efter 5000ms (5 sek.). Bemærk hvordan det hele startes af **body taggens onload event**.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Javascript eksempler</title>

<script type="text/javascript" >

   function start(){
      setTimeout('skift()', 5000) ;
   }

   function skift(){
         location.href='http://www.google.dk' ;
   }

</script>
</head>

<body onload="start()">

<h1>Hej - jeg "googler" videre</h1>

<script type="text/javascript" >

   window.status='Javascript var her';

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 4:

Dette eksempel viser at en eventhandler kan være en selvskrevet JavaScript funktion som tilfældet er her med **funktionen musclick.** Funktionen musclick anvender **window objektets open metode** til at åbne et nyt browservindue hvor i der vises indholdet af filen help.htm. Bemærk at det er muligt at bestemme vinduets størrelse og menuliniens (toolbar) udseende.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Javascript eksempler</title>

    <script type="text/javascript" >
        function musclick(){
            window.open('help.htm', '', 'toolbar=no, height=300, width=200');
        }
    </script>
</head>
<body>
    <h1>Åbning af webside i nyt vindue</h1>

    <h2 onclick="musclick();">Klik <i>her</i> for hjælp</h2>
</body>
</html>
```

JavaScript Programmering

Eksempel 5:

Dette eksempel viser at en variabel i JavaScript kan indeholde andet end almindelige værdier såsom tal og tekst. Først lader vi variablen **dw** være en reference til en ”anonym” funktion, der kalder document.write metoden, man siger at dw peger på eller refererer til funktionen.

Fordelen – ligetil – man kan give metoder mere ”programmør” venlige navne (kortere ☺)

Yderligere vises det hvordan man i JavaScript kan benytte det, at almindelige variable i virkeligheden er objekter med ”indbyggede” metoder, som er relevante for variabeltypen.

En variable (her et string objekt) har metoder til at give teksten farve, type og størrelse. Det skal dog bemærkes, at man ikke vil bruge disse metoder til styring af et dokuments udseende. Det overlader vi til css-styles!

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>

<body>
<h1>Variable som objekter og funktions reference</h1>

<script type="text/javascript" >

    var dw = function(str) { document.write(str) };
   var s='Killroy was here!';

   dw( s.fontcolor('#FF0000').italics().big() );

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 6:

Dette eksempel viser anvendelsen af Date objektet til at vise navnet på ugens dag. Da Date objektets **getDay metode** giver en numerisk værdi mellem 0 og 6, kan dennne værdi konverteres til et ugedags navn ved hjælp af et **array**, der netop indeholder disse navne.

```
 <html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>

<body>

<h1>Eksempel på dato og arrays</h1>
<script type="text/javascript" >

    var dw = function(str) { document.write(str) };
   var nu = new Date();  //variablen 'nu' indeholde dags dato/tid

    var
dagnavn=['Søndag','Mandag','Tirsdag','Onsdag','Torsdag','Fredag','Lørdag' ];

    dw( dagnavn[nu.getDay()]+', som er dag nr. ' + nu.getDay() );

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 7:

Dette eksempel viser anvendelsen af Date objektet til at vise dato og klokkeslæt.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>

<body>

<h1>Eksempel på dato og arrays</h1>
<script type="text/javascript" >

    var dw = function(str) { document.write(str) };
    var nu = new Date(); //variablen 'nu' indeholde dags dato/tid

    var dagnavn = ['søndag','mandag','tirsdag','onsdag','torsdag','fredag','lørdag'];
    var mdnavn = ['jan','feb','mar','apr','maj','jun','jul','aug','sep','okt','nov','dec'];

    dw(dagnavn[nu.getDay()] + ' den ' + nu.getDate() + '. ');
    dw(mdnavn[nu.getMonth()] + ' ' + nu.getFullYear());

    var tt = nu.getHours();
    var mm = nu.getMinutes();
    var ss = nu.getSeconds();

    if (tt < 10) { tt = '0' + tt; }
    if (mm < 10) { mm = '0' + mm; }
    if (ss < 10) { tt = '0' + ss; }

    dw(' klokken ' + tt + ':' + mm + ':' + ss)

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 8:

Dette eksempel viser anvendelsen af en funktion til at implementerer en **dwl** funktion**,** der indeholder et HTML linieskift (<br/>). Desuden vises de forskellige typer af variable som JavaScript kan håndtere. Samtidig vises anvendelsen af **typeof** operatoren. **BEMÆRK** at der anvendes dobbelt lighedstegn **==** ved sammenligning af værdier.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>

<body>

<h1>Eksempel på type undersøgelse</h1>
<script type="text/javascript" >

    var dwl = function(str) { document.write(str+'<br/>') };

   var i = 123;
   var f = 22.50;
   var b = true;
   var s = 'Glade jul';
   var a = [2,3];
   var d = new Date();

   dwl('Variablen "i" er en ' + typeof i );
   dwl('Variablen "f" er en ' + typeof f);
   dwl('Variablen "b" er en ' + typeof b);
   dwl('Variablen "s" er en ' + typeof s);
   dwl('Variablen "a" er en ' + typeof a);
   dwl('Variablen "d" er en ' + typeof d);

   if ((typeof i) == 'number') {
       dwl('variablen "i" er et tal');
   }
   else {
       dwl('variablen "i" er ikke et tal')
   }

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 9:

Her ses de 3 typer af ”message-bokse” i JavaScript; **confirm(’spørgsmål’)** der returnerer enten true eller false – afhængig af om brugeren svare bekræftende eller afkræftende til spørgsmålet, **prompt(’spørgsmål’, ’forslag’)** der returnerer den værdi som brugeren indtaster som svar på et spørgsmål. Til sidst har vi den mest brugte message-boks **alert(’besked’),** der viser beskeden i en boks – hvortil brugeren kun kan svarer OK.

Funktionen **parseInt(’tekst-med-heltal’)** konverterer tekst, hvis teksten består af cifre, til et heltal (integer). En tilsvarende funktion **parseFloat(’tekst-med-kommatal’)** konverterer til et kommatal (float).

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>

<body>
<script type="text/javascript" >

    var talA;
    var talB;
    var sum;

    if ( confirm('Ønsker du at fortsætte') ){
        talA = prompt('Indtast tal A','7');
        talB = prompt('Indtast tal B','9.5');

        if( isNaN(talA) ) {talA=0; }
        if( isNaN(talB) ) {talB=0; }

        sum = parseInt(talA) + parseFloat(talB)

        alert('Summen er:' + sum)
    }
    else{
        alert('Surt show!');
    }

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 10:

Her ses eksempler på de egenskaber en tag har, når den optræder som objekt. **innerHTML** der giver mulighed for at ændre indholdet af en tag, samt **style**, der giver mulighed for at bestemme al typografi omkring en ved hjælp af JavaScript.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>

  <style type="text/css">
      .lnk{cursor:pointer; color:Blue;}
   </style>

<script type="text/javascript">

    var gbi = function(id) { return document.getElementById(id) };

   function klik(){
      alert(gbi('gladtekst').innerHTML);
      gbi('gladtekst').innerHTML='Nej for der er <b>OVERARBEJDE</b>';
      gbi('gladtekst').style.backgroundColor='red';
      gbi('gladtekst').style.fontSize='42pt';

      alert('Pause!');
      gbi('gladtekst').style.display='none';
      alert('Pause!');
      gbi('gladtekst').style.display='block';
   }
</script>
</head>

<body>

<h2 id="gladtekst" >Jeg er så glad i dag, for nu har jeg snart fri!</h2>
<h2 class="lnk" onclick="klik();">Klik her</h2>

</body>
</html>
```

JavaScript Programmering

Eksempel 11:

Her ses metoden til at aflæse data fra en formulars input-felter, samt indskrive et resultat i et tilsvarende felt. Det der er hele humlen her; er at input felterne optræder som objekter, f.eks. document.forms['frmTest'].txttal1.value hvor **value egenskaben** har værdien af det, der står i feltet.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>
<body>

<script type="text/javascript">

   function beregn(frm){
        tal1= parseFloat(frm.txttal1.value);
        tal2= parseFloat(frm.txttal2.value);
        sum = tal1 + tal2;
        frm.txtsum.value=sum;
   }

</script>

<form name="frmTest" id="frmTest" action="">

Tal 1 <input type="text" name="txttal1" size="40" /><br/>
Tal 2 <input type="text" name="txttal2" size="40" /><br/>
Summen<input type="text" name="txtsum"  size="40" /><br/>
<input  type="button"
        name="cmdOK"
        value="Beregn"
        onclick="beregn(document.forms['frmTest'])" />

</form>

</body>
</html>
```

JavaScript Programmering

Eksempel 12:

Her ses hvordan man kan "tappe" et objekt for egenskaber og deres tilhørende værdi ved hjælp af strukturen **for(i in obj).** Bemærk at skriveformen navigator['appName'] giver det samme som navigator.appName, fordelen ved førstnævnte er, at det er muligt at erstatte 'appName' strengen med en streng-variabel og derved opnå en mere generel anvendelig kode.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>
</head>
<body>

<script type="text/javascript" >

   var i;
   for(i in navigator)
   {
      document.write(i+' = '+navigator[i]+'<br/>');
   }

  //if (navigator['appName']=='Microsoft Internet Explorer')
   if (navigator.appName == 'Microsoft Internet Explorer') {
       alert('Du bruger også MSIE');
   }
   else {
       alert('Mozilla haj??');
   }

</script>

</body>
</html>
```

JavaScript Programmering

Eksempel 13:

Her ses eksempler på de muligheder vi opnår med script styrede styles. Et mindre menu-system oprettes nemt – **bemærk** dette eksempel består af to sider.

Funktionen **mnuclick** vil skiftevis vise og skjule den aktuelle div tag (ses på næste side).

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Javascript eksempler</title>

    <style type="text/css">
      .showMe{display:block;}
      .hideMe{display:none;}
   </style>

<script type="text/javascript" >

    var gbi = function(id) { return document.getElementById(id) };
   function musind(obj){
      obj.style.color='blue';
      obj.style.cursor='pointer';
   }

   function musud(obj){
      obj.style.color='black';
      obj.style.cursor='default';
   }

   function mnuclick(obj){
       if (obj.className == 'showMe') {
         obj.className = 'hideMe';
       }
       else{
           obj.className = 'showMe';
      }
   }
</script>
</head>
```

JavaScript Programmering

Eksempel 13a:

Her ses resten af menusystem, her ses body'en hvor det kun er h3-taggene der er synlige. De underliggende div-tags har fået style classes med display=none, som fjerne elementerne fra websiden, først når display sættes til block, vil indholdet af div-taggen vises på websiden.

```
<body>

<h3 onmouseover="musind(this);" onmouseout="musud(this);"
    onclick="mnuclick( gbi('mnuA') );">Documentation</h3>
<div id="mnuA" class="hideMe">
HTML<br/>
ASP<br/>
JAVASCRIPT<br/>
</div>

<h3 onmouseover="musind(this);" onmouseout="musud(this);"
    onclick="mnuclick( gbi('mnuB') );">Information</h3>
<div id="mnuB" class="hideMe">
WEB<br/>
SEARCH<br/>
</div>

<h3 onmouseover="musind(this);" onmouseout="musud(this);"
    onclick="mnuclick( gbi('mnuC') );">Diverse</h3>
<div id="mnuC" class="hideMe">
EKSEMPLER<br/>
KURSUSBEVIS<br/>
</div>

</body>
</html>
```

JavaScript Programmering

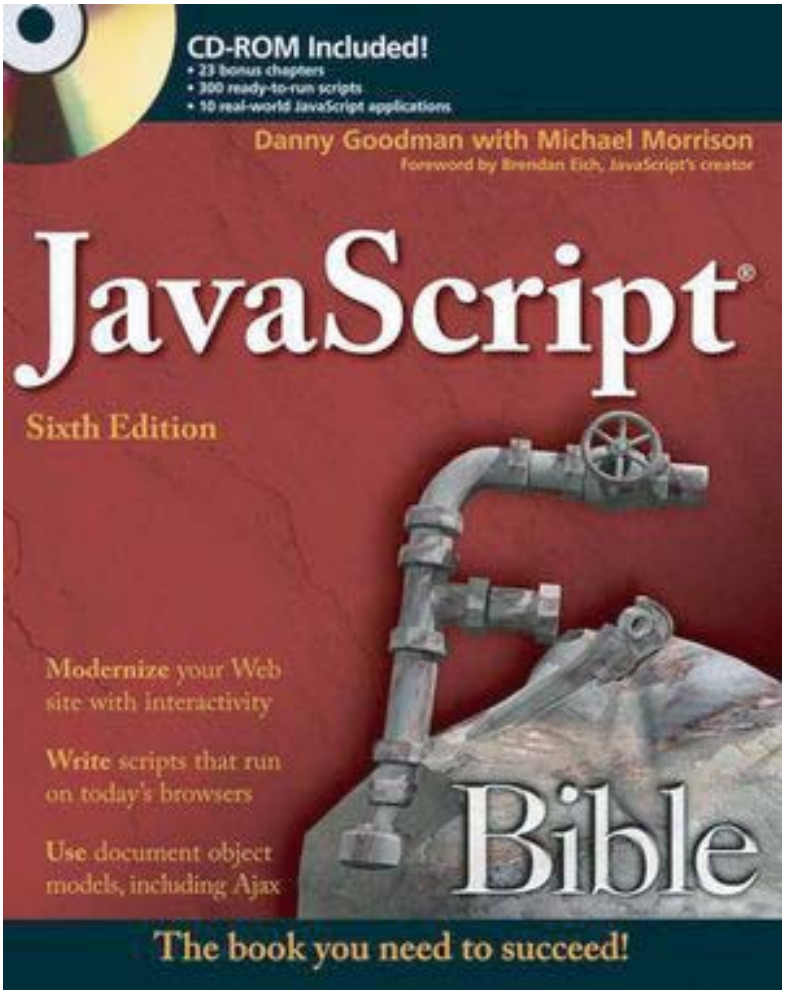
Eksempel 14:

Her ses et eksempel på muligheden for at skabe bevægelige elementer på websiden. Placeringen styres tidsmæssigt af en **intervaltimer**, der med 20ms intervaller kalder funktionen **Glide**, som flytter elementet 3 pixels til venstre indtil venstres kant nås. Herefter placeres elementet centreret på siden.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Javascript eksempler</title>

<script type="text/javascript">

var action;
var gbi = function(id) { return document.getElementById(id) };

function StartGlide(tagID) {
    //alert(document.body.offsetWidth+'px');
   gbi(tagID).style.left = document.body.offsetWidth+'px';
   gbi(tagID).style.visibility = "visible";
   action = window.setInterval("Glide ('" + tagID + "')", 20 );
}

function Glide(tagID){
    //
    gbi(tagID).style.left = (parseInt(gbi(tagID).style.left) - 3) + 'px';
   //alert(gbi(tagID).style.left);
   if ( parseInt(gbi(tagID).style.left) <=3){
      gbi(tagID).style.left=0+'px';
      gbi(tagID).style.position="static"
      gbi(tagID).style.textAlign="center";
      window.clearInterval( action );
   }
}

</script>

</head>
<body onload="StartGlide('glidetext');">
<div id="glidetext" style="position: absolute;font-size: 48pt ;color:
blue">Flyvende farmor</div>
</body>
</html>
```

JavaScript and Browser Objects Quick Reference

JavaScript Bible, 6th Edition
Danny Goodman
Appendix A



©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

String 28

constructor
length
prototype
anchor("*anchorName*")
big()
blink()
bold()
charAt(*index*)
charCodeAt([i])
concat(*string2*)
fixed()
fontcolor(#*rrggbb*)
fontsize(*1to7*)
fromCharCode(*n1*...)*
indexOf("*str*" [,i])
italics()
lastIndexOf("*str*" [,i])
link(*url*)
localeCompare()
match(*regexp*)
replace(*regexp*,*str*)
search(*regexp*)
slice(i,j)
small()
split(*char*)
strike()
sub()
substr(*start*,*length*)
substring(*intA*, *intB*)
sup()
toLocaleLowerCase()
toLocaleUpperCase()
toLowerCase()
toString()
toUpperCase()
valueOf()

*Method of the static String object.

Regular Expressions 42

global
ignoreCase
input
lastIndex
multiline
lastMatch
lastParen
leftContext
prototype
rightContext
source
$1...$9
compile(*regexp*)
exec("*string*")*
test("*string*")
str.match(*regexp*)
str.replace(*regexp*,"*string*")
str.search(*regexp*)
str.split(*regexp*[,*limit*])

*Returns array with properties: index, input, [0],...[n].

Array 31

constructor
length
prototype
concat(*array2*)
every(*func*[, *thisObj*])M1.8
filter(*func*[, *thisObj*])M1.8
forEach(*func*[, *thisObj*])M1.8
indexOf(*func*[, *thisObj*])M1.8
join("*char*")
lastIndexOf(*func*[, *thisObj*])M1.8
map(*func*[, *thisObj*])M1.8
pop()
push()
reverse()
shift()
slice(i,[j])
some(*func*[, *thisObj*])M1.8
sort(*compareFunc*)
splice(i,j[,*items*])
toLocaleString()
toString()
unshift()

Function 34

arguments
caller
constructor
length
prototype
apply(*this*, *argsArray*)
call(*this*[,*arg1*[,...*argN*]])
toString()
valueOf()

Date 30

constructor
prototype
getFullYear()
getYear()
getMonth()
getDate()
getDay()
getHours()
getMinutes()
getSeconds()
getTime()
getMilliseconds()
getUTCFullYear()
getUTCMonth()
getUTCDate()
getUTCDay()
getUTCHours()
getUTCMinutes()
getUTCSeconds()
getUTCMilliseconds()
parse("*dateString*")*
setYear(*val*)
setFullYear(*val*)
setMonth(*val*)
setDate(*val*)
setDay(*val*)
setHours(*val*)
setMinutes(*val*)
setSeconds(*val*)
setMilliseconds(*val*)
setTime(*val*)
setUTCFullYear(*val*)
setUTCMonth(*val*)
setUTCDate(*val*)
setUTCDay(*val*)
setUTCHours(*val*)
setUTCMinutes(*val*)
setUTCSeconds(*val*)
setUTCMilliseconds(*val*)
getTimezoneOffset()
toDateString()
toGMTString()
toLocaleDateString()
toLocaleString()
toLocaleTimeString()
toString()
toTimeString()
toUTCString()
UTC(*dateValues*)*

*Method of the static Date object.

Math* 29

E
LN2
LN10
LOG2E
LOG10E
PI
SQRT1_2
SQRT2
abs(*val*)
acos(*val*)
asin(*val*)
atan(*val*)
atan2(*val1*, *val2*)
ceil(*val*)
cos(*val*)
exp(*val*)
floor(*val*)
log(*val*)
max(*val1*, *val2*)
min(*val1*, *val2*)
pow(*val1*, *power*)
random()
round(*val*)
sin(*val*)
sqrt(*val*)
tan(*val*)

*All properties and methods are of the static Math object.

Error 32

prototype
constructor
descriptionE
fileNameE
lineNumber
message
name
numberE
toString()

Control Statements 32

```
if (condition) {
  statementsIfTrue
}

if (condition) {
  statementsIfTrue
} else {
  statementsIfFalse
}

result = condition ? expr1 : expr2

for ([init expr]; [condition]; [update expr]) {
  statements
}

for (var in object) {
  statements
}

for each ([var] varName in objectRef) {
   statements
}M1.8.1

with (objRef) {
   statements
}

do {
  statements
} while (condition)

yield valueM1.8.1

while (condition) {
  statements
}

return [value]

switch (expression) {
  case labelN :
     statements
     [break]
  ...
  [default :
     statements]
}

label :
continue [label]
break [label]

try {
  statements to test
}
catch (errorInfo) {
  statements if exception occurs in try block
}
[finally {
  statements to run, exception or not
}]

throw value
```

Number 29

constructor
MAX_VALUE
MIN_VALUE
NaN
NEGATIVE_INFINITY
POSITIVE_INFINITY
prototype
toExponential(n)
toFixed(n)
toLocaleString()
toString([*radix*])
toPrecision(n)
valueOf()

Boolean 29

constructor
prototype
toString()
valueOf()

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

Globals 35

Functions
atob()[M]
btoa()[M]
decodeURI("*encodedURI*")
decodeURIComponent("*encComp*")
encodeURI("*URIString*")
encodeURIComponent("*compString*")
escape("*string*" [,1])
eval("*string*")
isFinite(*number*)
isNaN(*expression*)
isXMLName("*string*")[M1.8.1]
Number("*string*")
parseFloat("*string*")
parseInt("string" [,*radix*])
toString([*radix*])
unescape("*string*")
unwatch(*prop*)
watch(*prop*, *handler*)

Statements
// /*...*/
const
var

Appendix A
JavaScript Bible, 6th Edition
by Danny Goodman

How to Use This Quick Reference

This guide contains quick reference info for the core JavaScript language and browser object models starting with IE 5.5, Mozilla, and Safari.

Numbers in the upper right corners of object squares are chapter numbers in which the object is covered in detail.

Each term is supported by all baseline browsers unless noted with a superscript symbol indicating browser brand and version:

E—Internet Explorer M—Mozilla S—Safari

For example, M1.4 means the term is supported only by Mozilla 1.4 or later; E means the terms is supported only by Internet Explorer.

Operators 33

Comparison

==	Equals
===	Strictly equals
!=	Does not equal
!==	Strictly does not equal
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

Arithmetic

+	Plus (and string concat.)
-	Minus
*	Multiply
/	Divide
%	Modulo
++	Increment
--	Decrement
-*val*	Negation

Assignment

=	Equals
+=	Add by value
-=	Subtract by value
*=	Multiply by value
/=	Divide by value
%=	Modulo by value
<<=	Left shift by value
>>=	Right shift by value
>>>=	Zero fill by value
&=	Bitwise AND by value
|=	Bitwise OR by value
^=	Bitwise XOR by value

Boolean

&&	AND
||	OR
!	NOT

Bitwise

&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Left shift
>>	Right shift
>>>	Zero fill right shift

Miscellaneous

,	Series delimiter
delete	Property destroyer
in	Item in object
instanceof	Instance of
new	Object creator
this	Object self-reference
typeof	Value type
void	Return no value

frameset 16

border	(None)	onload
borderColor[E]		
cols		
frameBorder[E]		
frameSpacing[E]		
rows		

iframe 16

align
allowTransparency[E]
contentDocument[MS]
contentWindow[EM]
frameBorder[E]
frameSpacing[E]
height
hspace[E]
longDesc
marginHeight
marginWidth
name
noResize
scrolling
src
vspace[E]
width

frame 16

allowTransparency[E]
borderColor[E]
contentDocument[MS]
contentWindow[EM]
frameBorder
height[E]
longDesc
marginHeight
marginWidth
name
noResize
scrolling
src
width[E]

popup[E] 16

document	hide()
isOpen	show()

location 17

hash	assign("*url*")
host	reload([*unconditional*])
hostname	replace("*url*")
href	
pathname	
port	
protocol	
search	

history 17

current[M(signed)]	back()
length	forward()
next[M(signed)]	go(*int* | "*url*")
previous[M(signed)]	

window 16

appCore[M]
clientInformation[ES1.2]
clipboardData[E]
closed
Components[][M]
content[M]
controllers[][M]
crypto[M]
defaultStatus
dialogArguments[E]
dialogHeight[E]
dialogLeft[E]
dialogTop[E]
dialogWidth[E]
directories[M]
document
event[ES]
external[E]
frameElement[EMS1.2]
frames[]
fullScreen[M1.4]
history
innerHeight[MS]
innerWidth[MS]
length
location
locationbar[M]
menubar[M]
name
navigator
netscape[M]
offscreenBuffering[ES1.2]
opener
outerHeight[MS]
outerWidth[MS]
pageXOffset[MS]
pageYOffset[MS]
parent
personalbar[M]
pkcs11[M]
prompter[M]
returnValue[E]
screen
screenLeft[ES1.2]
screenTop[ES1.2]
screenX[MS1.2]
screenY[MS1.2]
scrollbars[M]
scrollMaxX[M1.4]
scrollMaxY[M1.4]
scrollX[MS]
scrollY[MS]
self
sidebar[M]
status
statusbar[M]
toolbar[M]
top
window

addEventListener("*evt*", *func*, *capt*)[MS]
alert("*msg*")
attachEvent("*evt*", *func*)[E]
back()[M]
blur()
clearInterval(ID)
clearTimeout(ID)
close()
confirm("*msg*")
createPopup()[E]
detachEvent("*evt*", *func*)[E]
dispatchEvent()[MS]
dump("*msg*")[M1.4]
execScript("*exprList*"[, *lang*])[E]
find(["*str*"[, *case*[, *up*]]])[M]
fireEvent("*evt*"[, *evtObj*])[E]
focus()
forward()[M]
geckoActiveXObject(ID)[M1.4]
getComputedStyle(*node*, "")[M]
getSelection()[MS]
home()[M]
moveBy(Δx, Δy)
moveTo(x, y)
navigate("*url*")[E]
open("*url*", "*name*"[, *specs*])
openDialog("*url*", "*name*"[, *specs*])[M]
print()
prompt("*msg*", "*reply*")
removeEventListener("*evt*", *func*, *capt*)[MS]
resizeBy(Δx, Δy)
resizeTo(*width*, *height*)
scroll()
scrollBy(Δx, Δy)
scrollByLines(n)[M]
scrollByPages(n)[M]
scrollTo(x, y)
setInterval(*func*, *msecs*[, *args*])
setTimeout(*func*, *msecs*[, *args*])
showHelp("*url*")[E]
showModalDialog("*url*"[, *args*][, *features*])[ES2.01]
showModelessDialog("*url*"[, *args*][, *features*])[E]
sizeToContent()[M]
stop()[M]

onabort[M]
onafterprint[E]
onbeforeprint[E]
onbeforeunload[E]
onblur
onclick
onclose
onerror
onfocus
onhelp[E]
onkeydown
onkeypress
onkeyup
onload
onmousedown
onmousemove
onmouseout
onmouseover
onmouseup
onmove
onreset
onresize[EM]
onscroll[EMS1.3]
onunload

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

document 18

activeElement[E]
alinkColor
anchors[]
applets[]
baseURI[M]
bgColor
body
charset[E]
characterSet[M]
compatMode[EM]
contentType[M]
cookie
defaultCharset[E]
defaultView[M]
designMode[EM]
doctype[M]
documentElement
documentURI[M1.7]
domain
embeds[]
expando[E]
fgColor
fileCreatedDate[E]
fileModifiedDate[E]
fileSize[E]
forms[]
frames[]
height[MS]
images[]
implementation[E6MS]
inputEncoding[M1.8]
lastModified
linkColor
links[]
location
media[E]
mimeType[E]
nameProp[E6]
namespaces[]
parentWindow[E]
plugins[]
protocol[E]
referrer
scripts[][E]
security[E]
selection[E]
strictErrorChecking[M1.8]
styleSheets[]
title
URL
URLUnencoded[E]
vlinkColor
width[MS]
xmlEncoding[M1.8]
xmlStandalone[M1.8]
xmlVersion[M1.8]

clear()
close()
createAttribute(*"name"*)[E6MS]
createCDATASection(*"data"*)[M]
createComment(*"text"*)[E6MS]
createDocumentFragment()[E6MS]
createElement(*"tagname"*)
createElementNS(*"uri", "tagname*)
createEvent(*"evtType"*)[MS]
createEventObject([*evtObj*])[E]
createNSResolver(*nodeResolver*)[M]
createRange()[M]
createStyleSheet([*"url"*[, *index*]])[E]
createTextNode(*"text"*)
createTreeWalker(*root, what, filterfunc, exp*)[M1.4]
elementFromPoint(x, y)[E]
evaluate(*"expr", node, resolver, type, result*)[M]
execCommand(*"cmd"*[, *UI*][, *param*])[EM1.3S1.3]
getElementById(*"ID"*)
getElementsByName(*"name"*)
importNode(*node, deep*)[M]
open([*"mimetype"*][, *"replace"*])
queryCommandEnabled(*"commandName"*)[EM1.3]
queryCommandIndterm(*"commandName"*)
queryCommandState(*"commandName"*)
queryCommandSupported(*"commandName"*)
queryCommandText(*"commandName"*)
queryCommandValue(*"commandName"*)
recalc([*all*])[E]
write(*"string"*)
writeln(*"string"*)

onselectionchange[E]
onstop[E]

link 37

charset
disabled
href
hreflang[E6MS]
media
rel
rev
sheet[M]
styleSheet[E]
target
type

(None)

onload[E]

html 37

version[E6MS]

head 37

profile

title 37

text

base 37

href
target

meta 37

charset[E]
content
httpEquiv
name
url[E]

script 37

defer
event
htmlFor
src
text
type

All HTML Element Objects 15

accessKey
all[][E]
attributes[]
baseURI[M]
behaviorUrns[][E]
canHaveChildren[E]
canHaveHTML[E]
childNodes[]
children[ES1.2]
cite[E6MS]
className
clientHeight
clientLeft[E]
clientTop[E]
clientWidth
contentEditable[ES1.2]
currentStyle[E]
dateTime[E6M]
dataFld[E]
dataFormatAs[E]
dataSrc[E]
dir
disabled
document[ES1.2]
filters[][E]
firstChild
height
hideFocus[E]
id
innerHTML
innerText[ES]
isContentEditable[ES1.2]
isDisabled[E]
isMultiLine[E]
isTextEdit[E]
lang
language[E]
lastChild
length
localName[MS]
namespaceURI[MS]
nextSibling
nodeName
nodeType
nodeValue
offsetHeight
offsetLeft
offsetParent
offsetTop
offsetWidth
outerHTML[ES1.3]
outerText[ES1.3]
ownerDocument
parentElement[ES1.2]
parentNode
parentTextEdit[E]
prefix[MS]
previousSibling
readyState[E]
recordNumber[E]
runtimeStyle[E]
scopeName[E]
scrollHeight
scrollLeft
scrollTop
scrollWidth
sourceIndex[E]
style
tabIndex
tagName
tagUrn[E]
textContent[M1.7]
title
uniqueID[E]
unselectable[E]
width

addBehavior(*"url"*)[E]
addEventListener(*"evt", func,capt*)[MS]
appendChild(*node*)
applyElement(*elem*[, *type*])[E]
attachEvent(*"evt", func*)[E]
blur()
clearAttributes()[E]
click()
cloneNode(*deep*)
compareDocumentPosition(*node*)[M1.4]
componentFromPoint(x, y)[E]
contains(*elem*)[E]
createControlRange()[E]
detachEvent(*"evt", func*)[E]
dispatchEvent(*evtObj*)[MS]
doScroll(*"action"*)[E]
dragDrop()[E]
fireEvent(*"evtType"*[, *evtObj*])[E]
focus()
getAdjacentText(*"where"*)[E]
getAttribute(*"name"*[, *case*])
getAttributeNode(*"name"*)[E6MS]
getAttributeNodeNS(*"uri", "name"*)[M]
getAttributeNS(*"uri", "name"*)[M]
getBoundingClientRect()[E]
getClientRects()[E]
getElementsByTagName(*"tagname"*)
getElementsByTagNameNS(*"uri", "name"*)[M]
getExpression(*"attrName"*)[E]
getFeature(*"feature", "version"*)[M1.7.2]
getUserData(*"key"*)[M1.7.2]
hasAttribute(*"attrName"*)[MS]
hasAttributeNS(*"uri", "name"*)[M]
hasAttributes()[MS]
hasChildNodes()
insertAdjacentElement(*"where", obj*)[E]
insertAdjacentHTML(*"where", "HTML"*)[E]
insertAdjacentText(*"where", "text"*)[E]
insertBefore(*newNode, refNode*)
isDefaultNamespace(*"uri"*)[M1.7.2]
isEqualNode(*node*)[M1.7.2]
isSameNode(*node*)[M1.7.2]
isSupported(*"feature", "version"*)[MS]
item(*index*)
lookupNamespaceURI(*"prefix"*)[M1.7.2]
lookupPrefix(*"uri"*)[M1.7.2]
mergeAttributes(*srcObj*)[E]
normalize()
releaseCapture()[E]
removeAttribute(*"attrName"*[, *case*])
removeAttributeNode(*attrNode*)[E6MS]
removeAttributeNS(*"uri", "name"*)[M]
removeBehavior(*ID*)[E]
removeChild(node)
removeEventListener(*"evt", func,capt*)[MS]
removeExpression(*"propName"*)[E]
removeNode(*childrenFlag*)[E]
replaceAdjacentText(*"where", "text"*)[E]
replaceChild(*newNode, oldNode*)
replaceNode(*newNode*)[E]
scrollIntoView(*topFlag*)[EMS2.02]
setActive()[E]
setAttribute(*"name", "value"*[, *case*])
setAttributeNode(*attrNode*)[E6MS]
setAttributeNodeNS(*"uri", "name"*)[M]
setAttributeNS(*"uri", "name", "value"*)[M]
setCapture(*containerFlag*)[E]
setExpression(*"propName", "expr"*)[E]
setUserData(*"key", data, handler*)[M1.7.2]
swapNode(*nodeRef*)[E]
tags(*"tagName"*)[E]
toString()
urns(*"behaviorURN"*)[E]

onactivate[E]
onafterupdate[E]
onbeforecopy[ES1.3]
onbeforecut[ES1.3]
onbeforedeactivate[E]
onbeforeeditfocus[E]
onbeforepaste[ES1.3]
onbeforeupdate[E]
onblur
oncellchange[E]
onclick
oncontextmenu[EM]
oncontrolselect[E]
oncopy[ES1.3]
oncut[ES1.3]
ondataavailable[E]
ondatasetchanged[E]
ondatasetcomplete[E]
ondblclick
ondeactivate[E]
ondrag[ES1.3]
ondragend[ES1.3]
ondragenter[ES1.3]
ondragleave[ES1.3]
ondragover[ES1.3]
ondragstart[ES1.3]
ondrop[ES1.3]
onerrorupdate[E]
onfilterchange[E]
onfocus
onfocusin[E]
onfocusout[E]
onhelp[E]
onkeydown
onkeypress
onkeyup
onlayoutcomplete[E]
onlosecapture[E]
onmousedown
onmouseenter[E]
onmouseleave[E]
onmousemove
onmouseout
onmouseover
onmouseup
onmousewheel[E]
onmove[E]
onmoveend[E]
onmovestart[E]
onpaste[ES1.3]
onpropertychange[E]
onreadystatechange[EMS1.2]
onresize
onresizeend[E]
onresizestart[E]
onrowenter[E]
onrowexit[E]
onrowsdelete[E]
onrowsinserted[E]
onscroll[E]
onselectstart[ES1.3]

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

body 18

alink
background
bgColor
bgProperties[E]
bottomMargin[E]
leftMargin[E]
link
noWrap[E]
rightMargin[E]
scroll[E]
scrollLeft[EM]
scrollTop[EM]
text
topMargin[E]
vLink

createControlRange()[E]
createTextRange()[E]
doScroll(*"scrollAction"*)[E]

onafterprint[E]
onbeforeprint[E]
onscroll[E]

h1...h6 36

align

br 36

clear

blockquote, q 36

cite[E6MS]

font 36

color
face
size

Range[MS] 36

collapsed
commonAncestorContainer
endContainer
endOffset
startContainer
startOffset

cloneContents()
cloneRange()
collapse([*start*])
compareBoundaryPoints(*type,src)*
compareNode(*node*)
comparePoint(*node, offset*)
createContextualFragment("*text*")
deleteContents()
detach()
extractContents()
insertNode(*node*)
intersectsNode(*node*)
isPointInRange(*node*, off*offset*set)
selectNode(*node*)
selectNodeContents(*node*)
setEnd(*node,offset*)
setEndAfter(*node*)
setEndBefore(*node*)
setStart(*node,offset*)
setStartAfter(*node*)
setStartBefore(*node*)
surroundContents(*node*)
toString()

marquee 36

behavior[E]
bgColor[E]
direction[EM]
height[EM]
hspace[EM]
loop[E]
scrollAmount[EM]
scrollDelay[EM]
trueSpeed[E]
vspace[E]
width[E]

start()[EM]
stop()[EM]

onbounce[E]
onfinish[E]
onstart[E]

ol 38

start
type

ul 38

type

li 38

type
value

hr 36

align
color[E]
noShade
size
width

dl, dt, dd 38

compact

TextRange[E] 36

boundingHeight
boundingLeft
boundingTop
boundingWidth
htmlText
offsetLeft
offsetTop
text

collapse([*start*])
compareEndPoints("*type*",*range*)
duplicate()
execCommand("*cmd*"[,*UI*[,*val*]])
expand("*unit*")
findText("*str*"[,*scope,flags*])
getBookmark()
getBoundingClientRect()
getClientRects()
inRange(*range*)
isEqual(*range*)
move("*unit*"[,*count*])
moveEnd("*unit*"[,*count*])
moveStart("*unit*"[,*count*])
moveToBookmark("*bookmark*")
moveToElementText(*elem*)
moveToPoint(x,y)
parentElement()
pasteHTML("*HTMLText*")
queryCommandEnabled("*cmd*")
queryCommandIndeterm("*cmd*")
queryCommandState("*cmd*")
queryCommandSupported("*cmd*")
queryCommandText("*cmd*")
queryCommandValue("*cmd*")
scrollIntoView()
select()
setEndPoint("*type*",*range*)

canvas[M1.8S1.3] 20

fillStyle
globalAlpha
globalCompositeOperation
lineCap
lineJoin
lineWidth
miterLimit
shadowBlur
shadowColor
shadowOffsetX
shadowOffsetY
strokeStyle
target

arc(x, y, *radius*, *start*, *end*, *clockwise*)
arcTo(*x1*, *y1*, *x2*, *y2*, *radius*)
bezierCurveTo(*cp1x*, *cp1y*, *cp2x*, *cp2y*, x, y)
beginPath()
clearRect(x, y, *width*, *height*)
clip()
closePath()
createLinearGradient(*x1*, *y1*, *x2*, *y2*)
createPattern(*img*, *repetition*)
createRadialGradient(*x1*, *y1*, *radius1*, *x2*, *y2*, *radius2*)
drawImage(*img*, x, y)
drawImage(*img*, x, y, *width*, *height*)
fill()
fillRect(x, y, *width*, *height*)
getContext(*contextID*)
lineTo(x, y)
moveTo(x, y)
quadraticCurveTo(*cpx*, *cpy*, x, y)
rect(x, y, *width*, *height*)
restore()
rotate(*angle*)
save()
scale(x, y)
stroke()
strokeRect(x, y, *width*, *height*)
translate(x, y)

img 20

align
alt
border
complete[EM]
dynsrc[E]
fileCreatedDate[E]
fileModifiedDate[E]
fileSize[E]
fileUpdatedDate[E]
height
href
hspace
isMap
longDesc[E6MS]
loop[E]
lowsrc[EM]
mimeType[E6]
name
nameProp[E]
naturalHeight[M]
naturalWidth[M]
protocol[E]
src
start[E]
useMap
vspace
width
x[MS]
y[MS]

(None)

onabort
onerror
onload

a 19

charset[E6MS]
coords[E6MS]
hash
host
hostname
href
hreflang[E6MS]
Methods[E]
mimeType[E]
name
nameProp[E]
pathname
port
protocol
rel
rev
search
shape[E6MS]
target
type[E6MS]
urn[E]

selection 36

anchorNode[M]
anchorOffset[M]
focusNode[M]
focusOffset[M]
isCollapsed[M]
rangeCount[M]
type[E]
typeDetail[E]

addRange(range)[M]
clear()[E]
collapse(node, offset)[M]
collapseToEnd()[M]
collapseToStart()[M]
containsNode(node, entireFlag)[M]
createRange()[E]
deleteFromDocument()[M]
empty()[E]
extend(node, offset)[M]
getRangeAt(rangeIndex)[M]
removeAllRanges()[M]
removeRange(range)[M]
selectAllChildren(elementRef)[M]
toString()[M]

map 20

areas[]
name

(None)

onscroll[E]

area 20

alt
coords
hash
host
hostname
href
noHref
pathname
port
protocol
search
shape
target

TextRectangle[E] 36

bottom
left
right
top

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

form 21

acceptCharset
action
autocomplete[E]
elements[]
encoding[EM]
enctype[E6MS]
length
method
name
target

reset()
submit()

onreset
onsubmit

input 22/23/24

checked[(checkbox, radio)]
complete[(image)]
defaultChecked[(checkbox, radio)]
defaultValue[(text, password)]
form
maxLength[(text)]
name
readOnly[(text)]
size[(text)]
src[(image)]
type
value

select()[(text, password)]

onchange[(text)]

textarea 23

cols
form
name
readOnly
rows
type
value
wrap

createTextRange()
select()

onchange

select 24

form
length
multiple
name
options[]
options[i].defaultSelected
options[i].index
options[i].selected
options[i].text
options[i].value
selectedIndex
size
type
value

add(*newOption*[, *index*])[E]
add(*newOption*, *optionRef*)[MS]
remove(*index*)

onchange

option 24

defaultSelected
form
label
selected
text
value

fieldset, legend 21

align
form

label 21

form
htmlFor

optgroup[E6MS] 24

form
label

caption 38

align
vAlign

screen 39

availHeight
availLeft[MS]
availTop[MS]
availWidth
bufferDepth[E]
colorDepth
fontSmoothingEnabled[E]
height
pixelDepth
updateInterval[E]
width

table 38

align
background[E]
bgColor
border
borderColor[E]
borderColorDark[E]
borderColorLight[E]
caption
cellPadding
cells[E]
cellSpacing
cols[E]
datePageSize[E]
frame
height
rows
rules
summary[E6MS]
tbodies
tFoot
tHead
width

createCaption()
createTFoot()
createTHead()
deleteCaption()
deleteRow(i)
deleteTFoot()
deleteTHead()
firstPage()[E]
insertRow(i)
lastPage()
moveRow(*srcIndex*, *destIndex*)[E]
nextPage()[E]
previousPage()[E]
refresh()[E]

onscroll

tbody, tfoot, thead 38

align
bgColor
ch[E6MS]
chOff[E6MS]
rows
vAlign

deleteRow(i)
insertRow(i)
moveRow(*srcIndex*, *destIndex*)[E]

tr 38

align
bgColor
borderColor
borderColorDark
borderColorLight
cells
ch[E6MS]
chOff[E6MS]
height[E]
rowIndex
sectionRowIndex
vAlign

deleteCell(i)
insertCell(i)

td, th 38

abbr[E6MS]
align
axis[E6MS]
background[E]
bgColor
borderColor[E]
borderColorDark[E]
borderColorLight[E]
cellIndex
ch[E]
chOff[E]
colSpan
headers
height
noWrap
rowSpan
vAlign
width

col, colgroup 38

align
ch[E6MS]
chOff[E6MS]
span
vAlign
width

navigator 39

appCodeName
appMinorVersion[E]
appName
appVersion
browserLanguage[E]
cookieEnabled
cpuClass[E]
language[MS]
mimeTypes[MS]
onLine[E]
oscpu[MS]
platform
plugins[MS]
product[MS]
productSub[MS]
securityPolicy[M]
systemLanguage[E]
userAgent
userLanguage
userProfile[E]
vendor[MS]
vendorSub[MS]

javaEnabled()
preference(*name*[, *val*])[M(signed)]

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

event 25

altKey
altLeft[E]
behaviorCookie[E6]
behaviorPart[E]
bookmarks[E6]
boundElements[E6]
bubbles[MS]
button
cancelable[MS]
cancelBubble
charCode[MS]
clientX
clientY
contentOverflow[E]
ctrlKey
ctrlLeft[E]
currentTarget[MS]
dataFld[E6]
dataTransfer[ES2]
detail[MS2]
eventPhase[MS]
fromElement[E]
isChar[MS]
isTrusted[M1.7.5]
keyCode
layerX[MS]
layerY[MS]
metaKey[MS]
nextPage[E]
offsetX[E]
offsetY[E]
originalTarget[M]
pageX[MS]
pageY[MS]
propertyName[E]
qualifier[E6]
reason[E6]
recordset[E6]
relatedTarget[MS]
repeat[E]
returnValue[ES1.2]
saveType[E]
screenX
screenY
shiftKey
shiftLeft[E]
srcElement[ES1.2]
srcFilter[E]
srcUrn[E]
target[MS]
timeStamp[MS]
toElement[E]
type
view[MS]
wheelData[E]
x[E]
y[E]

initEvent()[MS]
initKeyEvent()[MS]
initMouseEvent()[MS]
initMutationEvent()[MS]
initUIEvent()[MS]
preventDefault()[MS]
stopPropagation()[MS]

XMLHttpRequest[EMS1.2] 27

readyState
responseText
responseXML
status
statusText

abort()
getAllResponseHeaders()
getResponseHeader("*headerName*")
open("*method*","*url*"[, *asyncFlag*])
send(*data*)
setRequestHeader("*name*","*value*")

onreadystatechange

applet 39

align
alt[E6MS]
altHTML[E]
archive[E6MS]
code
codeBase
height
hspace
name
object[E]
vspace
width
(Applet variables)

(Applet methods)

embed 39

align[M]
height[EM]
hidden[E]
name
pluginspage[M]
src[M]
units[M]
width[EM]
(Object variables)

mimeType[MS] 39

description
enabledPlugin
type
suffixes

plugin[MS] 39

name
filename
description
length

refresh()

object 39

align[ES]
alt[E6]
altHTML[E]
archive[E6MS]
baseHref[E]
baseURI[M]
border[E6MS]
classid[E]
code
codeBase
codeType
contentDocument[M]
data
declare[E6MS]
form
height
hspace
name
object[E]
standby[E6MS]
type
useMap[E6MS]
vspace
width
(Object variables)

(Object methods)

Notes

JavaScript Bible, 6th Edition. ©2007 Danny Goodman (dannyg.com). All Rights Reserved.

JavaScript and Browser Objects Quick Reference

style 26

Text & Fonts
color
font
fontFamily
fontSize
fontSizeAdjust[M]
fontStretch[M]
fontStyle
fontVariant[EMS1.3]
fontWeight
letterSpacing
lineBreak[E]
lineHeight
quotes[M]
rubyAlign[E]
rubyOverhang[E]
rubyPosition[E]
textAlign
textAlignLast[E]
textAutospace[E]
textDecoration
textDecorationBlink[E]
textDecorationLineThrough[E]
textDecorationNone[E]
textDecorationOverline[E]
textDecorationUnderline[E]
textIndent
textJustify[E]
textJustifyTrim[E]
textKashidaSpace[E]
textOverflow[E6S1.3]
textShadow[MS1.2]
textTransform
textUnderlinePosition[E]
unicodeBidi
whiteSpace
wordBreak[E]
wordSpacing[E6MS]
wordWrap[ES1.3]
writingMode[E]

Positioning
bottom
height
left
pixelBottom[ES]
pixelHeight[ES]
pixelLeft[ES]
pixelRight[ES]
pixelTop[ES]
pixelWidth[ES]
posBottom[E]
posHeight[E]
posLeft[E]
posRight[E]
posTop[E]
posWidth[E]
position
right
top
width
zIndex

Borders & Edges
border
borderBottom
borderLeft
borderRight
borderTop
borderBottomColor
borderLeftColor
borderRightColor
borderTopColor
borderBottomStyle
borderLeftStyle
borderRightStyle
borderTopStyle
borderBottomWidth
borderLeftWidth
borderRightWidth
borderTopWidth
borderColor
borderStyle
borderWidth
margin
marginBottom
marginLeft
marginRight
marginTop
outline[M1.8.1S1.2]
outlineColor[M1.8.1S1.2]
outlineStyle[M1.8.1S1.2]
outlineOffset[M1.8.1S1.2]
outlineWidth[M1.8.1S1.2]
padding
paddingBottom
paddingLeft
paddingRight
paddingTop

Tables
borderCollapse[EMS1.3]
borderSpacing
captionSide[MS]
emptyCells[MS1.3]
tableLayout

Lists
listStyle
listStyleImage
listStylePosition
listStyleType

Background
background
backgroundAttachment[EMS1.2]
backgroundColor
backgroundImage
gackgroundPosition
backgroundPositionX[ES1.3]
backgroundPositionY[ES1.3]
backgroundRepeat

Inline Display & Layout
clear
clip
clipBottom[E]
clipLeft[E]
clipRight[E]
clipTop[E]
content[MS1.3]
counterIncrement[M1.8]
counterReset[M1.8]
cssFloat[MS]
cursor[EMS1.3]
direction
display
filter[E]
layoutGrid[E]
layoutGridChar[E]
layoutGridLine[E]
layoutGridMode[E]
layoutGridType[E]
markerOffset[M]
marks[M]
maxHeight[E7MS]
maxWidth
minHeight
minWidth
MozOpacity[M]
opacity[M1.7.2S1.2]
overflow
overflowX[EM1.8S1.2]
overflowY[EM1.8S1.2]
styleFloat[E]
verticalAlign[EMS1.2]
visibility
width
zoom[E]

Printing
orphans[M]
widows[M]
page[M]
pageBreakAfter[EMS1.3]
pageBreakBefore[EMS1.3]
pageBreakInside[M]
size[M]

Miscellaneous
accelerator[E]
behavior[E]
cssText[EMS1.3]
imeMode[E]

Scrollbars
scrollbar3dLightColor[E]
scrollbarArrowColor[E]
scrollbarBaseColor[E]
scrollbarDarkShadowColor[E]
scrollbarFaceColor[E]
scrollbarHighlightColor[E]
scrollbarShadowColor[E]
scrollbarTrackColor[E]

styleSheet 26

cssRules[MS]
cssText[E]
disabled
href
id[E]
imports[E]
media
ownerNode[MS]
ownerRule[MS]
owningElement[E]
pages[E]
parentStyleSheet
readOnly[E]
rules
title
type

addImport("*url*"[, *index*])[E]
addRule("*selector*", "*spec*"[, *index*])[E]
deleteRule(*index*)[MS]
insertRule("*rule*", *index*)[MS]
removeRule(*index*)[E]

style (element) 26

media
type

cssRule, rule 26

cssText[MS]
parentStyleSheet[MS]
readOnly[E]
selectorText
style
type[MS]

Addison-Wesley's

JavaScript Reference Card

Kathleen M. Goelz and Carol J. Schwartz, Rutgers University

Javascript: A scripting language designed to be integrated into HTML code to produce enhanced, dynamic, interactive web pages.

DATA TYPES

Definition: The classification of values based on the specific categories in which they are stored.

Primitive Types: String, Boolean, Integer, Floating Point, Null, Void

Composite Types: Object, Array, Function. Composite data types are in separate sections of the code.

NUMERIC

Integer: Positive or negative numbers with no fractional parts or decimal places.

Floating Point: Positive or negative numbers that contain a decimal point or exponential notations.

String: A sequence of readable characters or text, surrounded by single or double quotes.

Boolean: The logical values True/False, etc. used to compare data or make decisions.

Null: The variable does not have a value; nothing to report. Null is not the same as zero, which is a numeric value.

Casting: Moving the contents of a variable of one type to a variable of a different type. You don't move the contents to a different variable; it stays in the same variable but the data type is changed or "re-cast".

VARIABLES

Definition: A placeholder for storing data. In JavaScript, a declaration statement consists of the reserved word **var** and the name (identifier) of one or more variables.

Format:

```
var variable_name
```

[**var** command is used to *declare* (create) variables]

Examples:

```
var myHouseColor
var myAddress
var vacation_house, condominium,
    primaryResidence
```

Rules for Naming Variables:

1. Variables cannot be reserved words.
2. Variables must begin with a letter or underscore and cannot begin with symbols, numbers, or arithmetic notations.
3. Spaces cannot be included in a variable name.

Hints:

1. Although variables in JavaScript can be used without being declared, it is good programming practice to declare (initialize), all variables.
2. Variable names are case sensitive; for example *X* does not equal *x*.

PEARSON
Addison Wesley

9 780321 320759 90000
ISBN 0-321-32075-1

INITIALIZING VARIABLES

Use the declaration statement to assign a value to the variable. The value is on the right of the equal sign; the variable is on the left.

Format:

```
var  variable_name = value
```

Examples:

```
var  myHouseColor = "yellow"
```

[literal string value yellow assigned to variable `myHouseColor`]

```
var  myAddress = 473
```

[numeric value 473 assigned to variable `myAddress`]

```
var  bookTitle = "Time Capsule", cost =
     28.95, publisher = "Tucker Bay"
```

[multiple variables can be assigned in one statement]

DECISION MAKING AND CONTROL STRUCTURES

Definition: Statements and structures used to change the order in which computer operations will occur.

Types:

Conditional Branching	IF, IF-ELSE, IF-ELSE IF, SWITCH, WHILE, DO, FOR

CONDITIONALS

IF Statement: A conditional branching statement used to determine whether a stated condition is TRUE.

Format:

```
if  (condition)   {
       statements if condition is TRUE
    }
```

Example:

```
if  (score >= 65")  {
       grade = "Pass";
       message = "Congratulations";
}
```

IF-ELSE Statement: A conditional branching statement that includes a path to follow if the condition is TRUE and a path to follow if the condition is FALSE.

Format:

```
if   (condition)   {
   statements if condition is TRUE;
}
else    {
   statements if condition is FALSE;
}
```

Example:

```
if  (score >= 65) {
   grade = "Pass";
   message = "Congratulations";
}
else     {
   grade = "Fail"
   message = "Try again";
}
```

IF-ELSE IF Statement: A conditional branching statement that allows for more than two possible paths. The first time a true condition is encountered, the statement is executed and the remaining conditions will not be tested.

Format:

```
if  (condition) {
   Statements if condition is TRUE;
}
else if (condition)  {
   Statements if condition is TRUE;
}
else  {
   Statements if no prior condition is
     true;
}
```

Example:

```
if        (score>=90)  {
   grade="A";
}
else if   (score>=80)  {
   grade="B";
}
else if   (score>=70)  {
   grade="C";
}
else if   (score>=65)  {
   grade="D";
}
else                   {
   grade="F";
}
```

SWITCH Statement: An alternative to the IF-ELSE IF statement for handling multiple options. Compares the expression to the test values to find a match.

Format:

```
switch (expression or variable name)  {
  case label:
    statements if expression matches
     this label;
    break;
  case label:
    statements if expression matches
     this label;
    break;
  default:
    statements if expression does not
     match any label;
    break;
  }
```

Example:

```
switch (colorchoice)        {
  case "red":
    document.bgColor="red";
    break;
  case "blue":
    document.bgColor="blue";
    break;
  default:
    document.bgColor="white";
    break;
  }
```

LOOPS

Loops cause a segment of code to repeat until a stated condition is met. You can use any loop format for any type of code

FOR LOOP:

Format:

```
For (intialize; conditional test;
     increment/decrement)         {
   Statements to execute;
}
```

Example:

```
For (var i=0; i<=10; i++)         {
   document.write ("This is line " + i);
}
```

DO/WHILE LOOP:

Format:

```
do    {
   Statements to execute;
}
while (condition);
```

Example:

```
var i=0;
do    {
   document.write ("This is line " + i);
   i++;
}
while (i <=10);
```

WHILE LOOP:

Format:

```
while (condition) {
   Statements;
   Increment/decrement;
}
```

Example:

```
var i = 0;
while (i<=10)     {
   document.write ("This is line " + i);
   i++;
}
```

Hint: Watch out for infinite loops, which do not have a stopping condition or have a stopping condition that will never be reached.

OBJECTS

Definition: Objects are a composite data type which contain properties and methods. JavaScript contains built-in objects and allows the user to create custom objects.

Creating Objects: Use the **new** constructor

```
var X = new Array()
```

Examples:

date, time, math, strings, arrays

ARRAY OBJECT

Definition: Array object is a variable that stores multiple values. Each value is given an index number in the array and each value is referred to by the array name and the index number. Arrays, like simple variables, can hold any kind of data. You can leave the size blank when you create an array. The size of the array will be determined by the number of items placed in it.

Format:

```
var arrayname = new Array(size)
```

Hint: When you create an array, you create a new instance of the array object. All properties and methods of the array object are available to your new array.

Example:

```
var days = new Array (7)
```

This creates an array of seven elements using the array constructor.

The first item is `days[0]`, the last item is `days[6]`.

Initializing Arrays:

Array items can be treated as simple variables:

```
days[0] = "Sunday";
days[1] = "Monday";
```

etc.

STRING OBJECT

Definition: String object is created by assigning a string to a variable, or by using the new object constructor.

Example:

```
var name = "Carol";
var name = new String("Carol");
```

Properties:

`Length:`	returns the number of characters in the string
`Prototype:`	allows the user to add methods and properties to the string

Methods:

String formatting methods (similar to HTML formatting tags)

```
String.big
String.blink
String.italics
```

Substring methods (allow user to find, match, or change patterns of characters in the string)

```
indexOf()
charAt()
replace()
```

MATH OBJECT

Definition: Math object allows arithmetic calculations not supported by the basic math operators. Math is a built-in object that the user does not need to define.

Examples:

`Math.abs(number)`	returns absolute value of the numeric argument
`Math.cos(number)`	returns the cosine of the argument, in radians
`Math.round(number)`	rounds number to the nearest integer

DATE/TIME OBJECTS

Date object provides methods for getting or setting information about the date and time.

Note: Dates before January 1, 1970 are not supported.

FUNCTIONS

Definition: A pre-written block of code that performs a specific task. Some functions return values; others perform a task like sorting, but return no value. Function names follow the same rules as variables names. There may or may not be an argument or parameter in the parenthesis, but the parenthesis has to be there.

User-defined Functions:

Example:

`ParseInt()` or `ParseFloat()` convert a string to a number.

To create a function:

Format:

```
function name_of_function (arguments) {
    statements to execute when
     function is called;
}
```

Example:

```
function kilosToPounds (){
    pounds=kilos*2.2046;
}
```

This new function takes the value of the variable kilos, multiplies it by 2.2046, and assigns the result to the variable pounds.

To call a function: Give the name of the function followed by its arguments, if any

`ParseInt(X);` converts the data stored in the variable X into a numeric value.

`kilosToPounds(17);` converts 17 kilos to the same mass in pounds, returning the value 37.4782.

METHODS

Definition: A special kind of function used to describe or instruct the way the object behaves. Each object type in JavaScript has associated methods available.

Examples:

```
array.sort();
document.write();
string.length();
```

Calling: To call or use a method, state the method name followed by its parameters in parentheses.

Example:

```
document.write("Hello, world!");
```

PUTTING IT TOGETHER: JAVASCRIPT AND HTML ON THE WEB

Cookies: Text-file messages stored by the browser on the user's computer

Purpose: To identify the user, store preferences, and present customized information each time the user visits the page

Types:

Temporary (transient, session) — stored in temporary memory and available only during active browser session

Persistent (permanent, stored) — remain on user's computer until deleted or expired

Browser Detection: A script written to determine which browser is running; determine if the browser has the capabilities to load the webpage and support the javascript code; and, if needed, load alternate javascript code to match the browser and platform.

Sniffing: A script written to determine whether a specific browser feature is present; i.e., detecting the presence of Flash before loading a webpage.

Event Handling: Use HTML event attributes (mouseover, mouse click, etc.) and connect event to a JavaScript function called an event handler

OPERATORS

ARITHMETIC

+	addition	adds two numbers
–	subtraction	subtracts one number from another
*	multiplication	multiplies two numbers
/	division	divides one number by another
%	modulus	returns the integer remainder after dividing two numbers
++	increment	adds one to a numeric variable
—	decrement	subtracts one from a numeric variable

STRING

+	concatenation	concatenates or joins two strings or other elements
+=	concatenation/ assignment	concatenates two string variables and assigns the result to the first variable

LOGICAL

&&	logical AND	Compares two operands; returns true if both are true, otherwise returns false
||	logical OR	Compares two operands; returns true if either operand is true, otherwise returns false
!	logical NOT	Returns false if its operand can be converted to true, otherwise returns false

COMPARISON

==	Returns true if the operands are equal
!=	Returns true if the operands are not equal
===	Returns true if the operands are equal and the same data type
!==	Returns true if the operands are not equal and/or not the same data type
>	Returns true if the first operand is greater than the second
>=	Returns true if the first operand is greater than or equal to the second
<	Returns true if the first operand is less than the second
<=	Returns true if the first operand is less than or equal to the second

ASSIGNMENT

=	Assigns the value of the seond operand to the first operand
+=	Adds two numeric operands and assigns the result to the first operand
–=	Subtracts the second operand from the first, and assigns the result to the first
*=	Multiplies two operands, assigns the result to the first
/=	Divides the first operand by the second, assigns the result to the first
%=	Finds the modulus of two numeric operands, and assigns the result to the first

RESERVED WORDS

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	shor	while
do	import	static	with
double	in	super	

<html>

Document Outline

<!DOCTYPE>	Version of (X)HTML
<html>	HTML document
<head>	Page information
<body>	Page contents

Comments

<!-- Comment Text -->

Page Information

<base />	Base URL
<meta />	Meta data
<title>	Title
<link />	Relevant resource
<style>	Style resource
<script>	Script resource

Document Structure

<h[1-6]>	Heading
	Page section
	Inline section
<p>	Paragraph
 	Line break
<hr />	Horizontal rule

Links

<a href="">	Page link
<a href="mailto:">	Email link
<a name="name">	Anchor
<a href="#name">	Link to anchor

Text Markup

	Strong emphasis
	Emphasis
<blockquote>	Long quotation
<q>	Short quotation
<abbr>	Abbreviation
<acronym>	Acronym
<address>	Address
<pre>	Pre-formatted text
<dfn>	Definition
<code>	Code
<cite>	Citation
<del>	Deleted text
<ins>	Inserted text
<sub>	Subscript
<sup>	Superscript
<bdo>	Text direction

Lists

	Ordered list
	Unordered list
	List item
<dl>	Definition list
<dt>	Definition term
<dd>	Term description

Forms

<form>	Form
<fieldset>	Collection of fields
<legend>	Form legend
<label>	Input label
<input />	Form input
<select>	Drop-down box
<optgroup>	Group of options
<option>	Drop-down options
<textarea>	Large text input
<button>	Button

Tables

<table>	Table
<caption>	Caption
<thead>	Table header
<tbody>	Table body
<tfoot>	Table footer
<colgroup>	Column group
<col />	Column
<tr>	Table row
<th>	Header cell
<td>	Table cell

Images and Image Maps

<img />	Image
<map>	Image Map
<area />	Area of Image Map

Common Character Entities

"	"	Quotation mark
&	&	Ampersand
<	<	Less than
>	>	Greater than
@	@	"At" symbol
€	€	Euro
•	•	Small bullet
™	™	Trademark
£	£	Pound
		Non-breaking space
©	©	Copyright symbol

Objects

<object>	Object
<param />	Parameter

Empty Elements

<area />	<img />
<base />	<input />
 	<link />
<col />	<meta />
<hr />	<param />

Core Attributes

class	style
id	title

Note: Core Attributes may not be used in base, head, html, meta, param, script, style or title elements.

Language Attributes

dir	lang

Note: Language Attributes may not be used in base, br, frame, frameset, hr, iframe, param or script elements.

Keyboard Attributes

accesskey	tabindex

Window Events

onLoad	onUnload

Form Events

onBlur	onReset
onChange	onSelect
onFocus	onSubmit

Keyboard Events

onKeydown	onKeyup
onKeypress	

Mouse Events

onClick	onMouseout
onDblclick	onMouseover
onMousedown	onMouseup
onMousemove	

Available free from AddedBytes.com

Cascading Style Sheets (CSS 2)

BACKGROUND	
background	*background-color* *background-image* *background-repeat* *background-attachment* *background-position*
background-attachment	scroll | fixed
background-color	*color-rgb* *color-hex* *color-name* transparent
background-image	*url* none
background-position	top left | top center | top right | center left | center center | center right | bottom left | bottom center | bottom right *x-% y-%* *x-pos y-pos*
background-repeat	repeat | repeat-x | repeat-y | no-repeat

BORDER	
border	*border-width* *border-style* *border-color*
border-bottom	*border-bottom-width* *border-style* *border-color*
border-bottom-color	*border-color*
border-bottom-style	*border-style*
border-bottom-width	thin | medium | thick *length*
border-color	*color*
border-left	*border-left-width* *border-style* *border-color*
border-left-color	*border-color*
border-left-style	*border-style*
border-left-width	thin | medium | thick *length*
border-right	*border-right-width* *border-style* *border-color*
border-right-color	*border-color*
border-right-style	*border-style*
border-right-width	thin | medium | thick *length*
border-style	none | hidden | dotted | dashed | solid | double | groove | ridge | inset | outset
border-top	*border-top-width* *border-style* *border-color*
border-top-color	*border-color*
border-top-style	*border-style*
border-top-width	thin | medium | thick *length*
border-width	thin | medium | thick *length*

Values in italics are place holders for an actual value (like 1px, 1em, 1%), values in normal text are values that can be used as the actual value

CLASSIFICATION	
clear	left | right | both | none
cursor	*url* auto | crosshair | default | pointer | move | e-resize | ne-resize | nw-resize | n-resize | se-resize | sw-resize | s-resize | w-resize | text | wait | help
display	none | inline | block | list-item | run-in | compact | marker | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption
float	left | right | none
position	static | relative | absolute | fixed
visibility	visible | hidden | collapse

DIMENSION	
height	auto *length* *%*
line-height	normal *number* *length* *%*
max-height	none *length* *%*
max-width	none *length* *%*
min-height	*length* *%*
min-width	*length* *%*
width	auto *%* *length*

FONT	
font	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption | icon | menu | message-box | small-caption | status-bar
font-family	*family-name* *generic-family*
font-size	xx-small | x-small | small | medium | large | x-large | xx-large | smaller | larger *length* *%*
font-size-adjust	none *number*
font-stretch	normal | wider | narrower | ultra-condensed | extra-condensed | condensed | semi-condensed | semi-expanded | expanded | extra-expanded | ultra-expanded
font-style	normal | italic | oblique
font-variant	normal | small-caps
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

GENERATED CONTENT	
content	*string* *url* counter(*name*) counter(*name, list-style-type*) counters(*name, string*) counters(*name, string, list-style-type*) attr(X) open-quote | close-quote | no-open-quote | no-close-quote
counter-increment	none *identifier number*
counter-reset	none *identifier number*
quotes	none *string string string string*

LIST & MARKERS	
list-style	*list-style-type* *list-style-position* *list-style-image*
list-style-image	none *url*
list-style-position	Inside | outside
list-style-type	none | disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-alpha | upper-alpha | lower-greek | lower-latin | upper-latin | hebrew | armenian | georgian | cjk-ideographic | hiragana | katakana | hiragana-iroha | katakana-iroha
marker-offset	auto *length*

MARGIN	
margin	*margin-top* *margin-right* *margin-bottom* *margin-left*
margin-bottom	auto *length* *%*
margin-left	auto *length* *%*
margin-right	auto *length* *%*
margin-top	auto *length* *%*

OUTLINE	
outline	*outline-color* *outline-style* *outline-width*
outline-color	*color* invert
outline-style	None | dotted | dashed | solid | double | groove | ridge | inset | outset
outline-width	thin | medium | thick *length*

CSS Properties in **Dark Red** are shorthand properties and each value must be defined. The exception is where the property can define from one to four of the sides of a box element property (Top-Right-Bottom-Left) - i.e. border-width

©2008 Veign, All Rights Reserved

PADDING	
padding	*padding-top* *padding-right* *padding-bottom* *padding-left*
padding-bottom	*length* *%*
padding-left	*length* *%*
padding-right	*length* *%*
padding-top	*length* *%*
POSITIONING	
bottom	auto *%* *length*
clip	*shape* auto
left	auto *%* *length*
overflow	visible | hidden | scroll | auto
position	static | relative | absolute | fixed
right	auto *%* *length*
top	auto *%* *length*
vertical-align	Baseline | sub | super | top | text-top | middle | bottom | text-bottom *length* *%*
z-index	auto

UNITS	
MEASUREMENT	
%	percentage
cm	centimeter
em	1em = current font size of current element
ex	1ex = ~1/2 current size of current element
in	inch
mm	millimeter
pc	pica (= 12 points)
pt	point (= 1/72 inch)
px	pixel
COLORS	
Color name	Red, blue, green, dark green
rgb(x,y,z)	Red = rgb(255,0,0)
rgb(x%,y%,z%)	Red = rgb(100%,0,0)
#rrggbb	Red = #ff0000 (or shorthand = #f00)

TABLE	
border-collapse	collapse | separate
border-spacing	*length length*
caption-side	top | bottom | left | right
empty-cells	show | hide
table-layout	auto | fixed
TEXT	
color	*color*
direction	ltr | rtl
letter-spacing	normal *length*
text-align	left | right | center | justify
text-decoration	none | underline | overline | line-through | blink
text-indent	*length* *%*
text-shadow	none *color* *length*
text-transform	none | capitalize | uppercase | lowercase
unicode-bidi	normal | embed | bidi-override
white-space	normal | pre | nowrap
word-spacing	normal *length*

PSEUDO-CLASS	
:active	Adds special style to an activated element
:focus	Adds special style to an element while the element has focus
:hover	Adds special style to an element when you mouse over it
:link	Adds special style to an unvisited link
:visited	Adds special style to a visited link
:first-child	Adds special style to an element that is the first child of some other element
:lang	Allows the author to specify a language to use in a specified
PSEUDO-ELEMENT	
:first-letter	Adds special style to the first letter of a text
:first-line	Adds special style to the first line of a text
:before	Inserts some content before an element
:after	Inserts some content after an element

SELECTOR TYPES		
Name	**Info**	**Example**
Universal	Any element	* { font: 10px Arial; }
Type	Any element of that type	h1 { text-decoration: underline; }
Grouping	Multiple elements of different types	h1, h2, h3 { font-family: Verdana; }
Class	Multiple elements of different types when you don't want to affect all instances	.sampleclass { text-decoration: underline; }
Id	A single element type when you don't want to affect all instances of	#sampleid { text-decoration: underline; }
Descendant	An element that is below (in the document tree) another element—no matter how many levels below	#gallery h1 { text-decoration: underline; }
Child	An element that is directly below (in the document tree) another	#title > p { font-weight: bold; }
Sibling	All elements of a type that share the same parent	h1 + p { font-style: italic; }

Attribute	An element with that matches the attribute listed	option[selected] { color: #ff0000; }
		input[type="pass"] { color: #ccc; }
		img[src="sm.gif"] { border: 1px solid #000; }
		a[rel~="next"] { color: #fff; }
		*[lang|="en"] { color : red; }

SELECTOR PATTERNS	
Name	**Example**
*****	any element
E	an element of type E
E[foo]	an E element with a "foo" attribute
E[foo="bar"]	an E element whose "foo" attribute value is exactly equal to "bar"
E[foo~="bar"]	an E element whose "foo" attribute value is a list of space-separated values, one of which is exactly equal to "bar"
E [hreflang|="en"]	an E element whose "hreflang" attribute has a hyphen-separated list of values beginning (from the left) with "en"
E:first-child	an E element, first child of its parent
E:link **E:visited**	an E element being the source anchor of a hyperlink of which the target is not yet visited (:link) or already visited (:visited)
E:active **E:hover** **E:focus**	an E element during certain user actions
E:lang(fr)	an element of type E in language "fr" (the document language specifies how language is determined)
E::first-line	the first formatted line of an E element
E::first-letter	the first formatted letter of an E element
E::before	generated content before an E element
E::after	generated content after an E element
E.warning	an E element whose class is "warning" (the document language specifies how class is determined).
E#myid	an E element with ID equal to "myid".
E F	an F element descendant of an E element
E > F	an F element child of an E element
E + F	an F element immediately preceded by an E element

©2008 Veign, All Rights Reserved

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

✻ CORE

jQuery.holdReady(hold) *u*

Holds or releases the execution of jQuery's ready event.

jQuery() *jQ*

Accepts a string containing a CSS selector which is then used to match a set of elements.

jQuery(selector [, context]) - jQuery(element)
jQuery(object) - jQuery(elementArray)
jQuery(jQuery object)

jQuery.noConflict([removeAll]) *Obj*

Relinquish jQuery's control of the $ variable.

jQuery.sub() *jQ*

Creates a new copy of jQuery whose properties and methods can be modified without affecting the original jQuery object.

jQuery.when(deferreds) *pr*

Provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events.

✻ SELECTORS

All Selector ("*")

Selects all elements.

:animated Selector

Select all elements that are in the progress of an animation at the time the selector is run.

Attribute Contains Prefix Selector [name|="value"]

Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-).

Attribute Contains Selector [name*="value"]

Selects elements that have the specified attribute with a value containing the a given substring.

Attribute Contains Word Selector [name~="value"]

Selects elements that have the specified attribute with a value containing a given word, delimited by spaces.

Attribute Ends With Selector [name $="value"]

Selects elements that have the specified attribute with a value ending exactly with a given string. The comparison is case sensitive.

Attribute Equals Selector [name="value"]

Selects elements that have the specified attribute with a value exactly equal to a certain value.

Attribute Not Equal Selector [name!="value"]

Select elements that either don't have the specified attribute, or do have the specified attribute but not with a certain value.

Attribute Starts With Selector [name^="value"]

Selects elements that have the specified attribute with a value beginning exactly with a given string.

:button Selector

Selects all button elements and elements of type button.

:checkbox Selector

Selects all elements of type checkbox.

:checked Selector

Matches all elements that are checked.

Child Selector ("parent > child")

Selects all direct child elements specified by "child" of elements specified by "parent".

Class Selector (".class")

Selects all elements with the given class.

:contains() Selector

Select all elements that contain the specified text.

Descendant Selector ("ancestor descendant")

Selects all elements that are descendants of a given ancestor.

:disabled Selector

Selects all elements that are disabled.

Element Selector ("element")

Selects all elements with the given tag name.

:empty Selector

Select all elements that have no children (including text nodes).

:enabled Selector

Selects all elements that are enabled.

:eq() Selector

Select the element at index n within the matched set.

:even Selector

Selects even elements, zero-indexed.

:file Selector

Selects all elements of type file.

:first-child Selector

Selects all elements that are the first child of their parent.

:first Selector

Selects the first matched element.

:focus selector

Selects element if it is currently focused.

:gt() Selector

Select all elements at an index greater than index within the matched set.

Has Attribute Selector [name]

Selects elements that have the specified attribute, with any value.

:has() Selector

Selects elements which contain at least one element that matches the specified selector.

:header Selector

Selects all elements that are headers, like h1, h2, h3 and so on.

:hidden Selector

Selects all elements that are hidden.

ID Selector ("#id")

Selects a single element with the given id attribute.

:image Selector

Selects all elements of type image.

:input Selector

Selects all input, textarea, select and button elements.

:last-child Selector

Selects all elements that are the last child of their parent.

:last Selector

Selects the last matched element.

:lt() Selector

Select all elements at an index less than index within the matched set.

Multiple Attribute Selector [name="value"][name2="value2"]

Matches elements that match all of the specified attribute filters.

Multiple Selector ("selector1, selector2, selectorN")

Selects the combined results of all the specified selectors.

Next Adjacent Selector ("prev + next")

Selects all next elements matching "next" that are immediately preceded by a sibling "prev".

Next Siblings Selector ("prev ~ siblings")

Selects all sibling elements that follow after the "prev" element, have the same parent, and match the filtering "siblings" selector.

:not() Selector

Selects all elements that do not match the given selector.

:nth-child() Selector

Selects all elements that are the nth-child of their parent.

:odd Selector

Selects odd elements, zero-indexed.

:only-child Selector

Selects all elements that are the only child of their parent.

:parent Selector

Select all elements that are the parent of another element, including text nodes.

:password Selector

Selects all elements of type password.

:radio Selector

Selects all elements of type password.

:reset Selector

Selects all elements of type reset.

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

:selected Selector

Selects all elements that are selected.

:submit Selector

Selects all elements of type submit.

:text Selector

Selects all elements of type text.

:visible Selector

Selects all elements that are visible.

✻ ATTRIBUTES

.addClass() *jQ*

Adds the specified class(es) to each of the set of matched elements.

.addClass(className) - .addClass(function(index, currentClass))

.attr() *Str*

Get the value of an attribute for the first element in the set of matched elements.

.attr(attributeName) - .attr(attributeName, value)

.hasClass(className) *Str*

Determine whether any of the matched elements are assigned the given class.

.html() *Str* *jQ*

Get the HTML contents of the first element in the set of matched elements.

.html(htmlString) - .html(function(index, oldhtml))

.prop() *Str* *jQ*

Get the value of a property for the first element in the set of matched elements.

.prop(propertyName) - .prop(propertyName, value)

★ .removeAttr(attributeName) *jQ*

Remove an attribute from each element in the set of matched elements.

.removeClass() *jQ*

Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.removeProp(propertyName) *jQ*

Remove a property for the set of matched elements.

.toggleClass(className) *jQ*

Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.val() *Str* *N* *aO* *jQ*

Get the current value of the first element in the set of matched elements.

.val(value) - .val(function(index, value))

✻ TRAVERSING

.add() *jQ*

Add elements to the set of matched elements.

.add(selector) - .add(elements) - .add(html)
.add(jQuery object) - .add(selector, context)

.andSelf() *jQ*

Add the previous set of elements on the stack to the current set.

.children([selector]) *jQ*

Get the children of each element in the set of matched elements, optionally filtered by a selector.

.closest() *jQ* *aO*

Get the first element that matches the selector, beginning at the current element and progressing up through the DOM tree.

.closest(selector) - .closest(selector [, context])
.closest(jQuery object) - .closest(element)
.closest(selectors [, context])

.contents() *jQ*

Get the children of each element in the set of matched elements, including text and comment nodes.

.each(function(index, Element)) *jQ*

Iterate over a jQuery object, executing a function for each matched element.

.end() *jQ*

End the most recent filtering operation in the current chain and return the set of matched elements to its previous state.

.eq(index) *jQ*

Reduce the set of matched elements to the one at the specified index.

.filter() *jQ*

Reduce the set of matched elements to those that match the selector or pass the function's test.

.filter(selector) - .filter(function(index))
.filter(element) - .filter(jQuery object)

.find(selector) *jQ*

Get the descendants of each element in the current set of matched elements, filtered by a selector, jQuery object, or element.

.find(selector) - .find(jQuery object) - .find(element)

.first() *jQ*

Reduce the set of matched elements to the first in the set.

.has() *jQ*

Reduce the set of matched elements to those that have a descendant that matches the selector or DOM element.

.has(selector) - .has(contained)

★ .is() *0-1*

Check the current matched set of elements against a selector, element, or jQuery object and return true if at least one of these elements matches the given arguments.

.is(selector) - .is(function(index)) - .is(jQuery object)
.is(element)

.last() *jQ*

Reduce the set of matched elements to the final one in the set.

.map(callback(index, domElement)) *jQ*

Pass each element in the current matched set through a function, producing a new jQuery object containing the return values.

.next([selector]) *jQ*

Get the immediately following sibling of each element in the set of matched elements. If a selector is provided, it retrieves the next sibling only if it matches that selector.

.nextAll([selector]) *jQ*

Get all following siblings of each element in the set of matched elements, optionally filtered by a selector.

.nextUntil() *jQ*

Get all following siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object passed.

.nextUntil([selector] [, filter]) - .nextUntil([element] [, filter])

.not() *jQ*

Remove elements from the set of matched elements.

.not(selector) - .not(elements) - .not(function(index))

.offsetParent() *jQ*

Get the closest ancestor element that is positioned.

.parent([selector]) *jQ*

Get the parent of each element in the current set of matched elements, optionally filtered by a selector.

.parents([selector]) *jQ*

Get the ancestors of each element in the current set of matched elements, optionally filtered by a selector.

.parentsUntil() *jQ*

Get the ancestors of each element in the current set of matched elements, up to but not including the element matched by the selector, DOM node, or jQuery object.

.parentsUntil([selector] [, filter])
.parentsUntil([element] [, filter])

.prev([selector]) *jQ*

Get the immediately preceding sibling of each element in the set of matched elements, optionally filtered by a selector.

.prevAll([selector]) *jQ*

Get all preceding siblings of each element in the set of matched elements, optionally filtered by a selector.

.prevUntil() *jQ*

Get all preceding siblings of each element up to but not including the element matched by the selector, DOM node, or jQuery object.

.prevUntil([selector] [, filter])
.prevUntil([element] [, filter])

.siblings([selector]) *jQ*

Get the siblings of each element in the set of matched elements, optionally filtered by a selector.

.slice(start [, end]) *jQ*

Reduce the set of matched elements to a subset specified by a range of indices.

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

✻ MANIPULATION

.addClass() *jQ*

Adds the specified class(es) to each of the set of matched elements.

.addClass(className) - .addClass(function(index, currentClass))

.after() *jQ*

Insert content, specified by the parameter, after each element in the set of matched elements.

.after(content [, content]) - .after(function(index))

.append() *jQ*

Insert content, specified by the parameter, to the end of each element in the set of matched elements.

.append(content [, content])
.append(function(index, html))

.appendTo(target) *jQ*

Insert every element in the set of matched elements to the end of the target.

.attr() *jQ*

Get the value of an attribute for the first element in the set of matched elements.

.attr(attributeName) - .attr(attributeName, value)

.before() *jQ*

Insert content, specified by the parameter, before each element in the set of matched elements.

.before(content [, content]) - .before(function)

.clone() *jQ*

Create a deep copy of the set of matched elements.

.clone([withDataAndEvents])
.clone([withDataAndEvents] [,deepWithDataAndEvents])

.css() *Str* *jQ*

Get the value of a style property for the first element in the set of matched elements.

.css(propertyName) - .css(propertyName, value)
.css(propertyName, function(index, value))
.css(map)

.detach([selector]) *jQ*

Remove the set of matched elements from the DOM.

.empty() *jQ*

Remove all child nodes of the set of matched elements from the DOM.

.hasClass(className) *Str*

Determine whether any of the matched elements are assigned the given class.

.html() *Str* *jQ*

Get the HTML contents of the first element in the set of matched elements.

.html(htmlString) - .html(function(index, oldhtml))

.innerHeight() *Int*

Get the current computed height for the first element in the set of matched elements, including padding but not border.

.innerWidth() *Int*

Get the current computed width for the first element in the set of matched elements, including padding but not border.

.insertAfter(target) *Int*

Insert every element in the set of matched elements after the target.

.insertBefore(target) *Int*

Insert every element in the set of matched elements before the target.

.offset() *Int*

Get the current coordinates of the first element in the set of matched elements, relative to the document.

.offset(coordinates) - .offset(function(index, coords))

.outerHeight([includeMargin]) *Int*

Get the current computed height for the first element in the set of matched elements, including padding, border, and optionally margin. Returns an integer (without "px") representation of the value or null if called on an empty set of elements.

.position() *Obj*

Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

.prepend(content [, content]) *jQ*

Insert content, specified by the parameter, to the beginning of each element in the set of matched elements.

.prependTo(target) *jQ*

Insert every element in the set of matched elements to the beginning of the target.

.prop() *Str* *jQ*

Get the value of a property for the first element in the set of matched elements.

.prop(propertyName) - .prop(propertyName, value)

.remove([selector]) *Str*

Remove the set of matched elements from the DOM.

.removeAttr(attributeName) *jQ*

Remove an attribute from each element in the set of matched elements.

.removeClass() *jQ*

Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.removeProp(propertyName) *jQ*

Remove a property for the set of matched elements.

.replaceAll(target) *jQ*

Replace each target element with the set of matched elements.

.replaceWith() *jQ*

Replace each element in the set of matched elements with the provided new content.

.replaceWith(newContent) - .replaceWith(function)

.scrollLeft() *jQ*

Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

.scrollLeft(value)

.scrollTop() *jQ*

Get the current vertical position of the scroll bar for the first element in the set of matched elements.

.scrollTop value)

.text() *jQ*

Get the combined text contents of each element in the set of matched elements, including their descendants.

.text(textString) - .text(function(index, text))

.toggleClass() *jQ*

Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.toggleClass(className)
.toggleClass(className, switch)

.unwrap() *jQ*

Remove the parents of the set of matched elements from the DOM, leaving the matched elements in their place.

.val() *Str* *N* *a()* *jQ*

Get the current value of the first element in the set of matched elements.

.val(value) - .val(function(index, value))

.width() *Int*

Get the current computed width for the first element in the set of matched elements.

.width(value) - .width(function(index, width))

.wrap() *jQ*

Wrap an HTML structure around each element in the set of matched elements.

.wrap(wrappingElement) - .wrap(function(index))

.wrapAll(wrappingElement) *jQ*

Wrap an HTML structure around all elements in the set of matched elements.

.wrapInner() *jQ*

Wrap an HTML structure around the content of each element in the set of matched elements.

.wrapInner(wrappingElement)
.wrapInner(function(index))

✻ CSS

.addClass() *jQ*

Adds the specified class(es) to each of the set of matched elements.

.addClass(className) - .addClass(function(index, currentClass))

.css() *Str* *jQ*

Get the value of a style property for the first element in the set of matched elements.

.css(propertyName) - .css(propertyName, value)
.css(propertyName, function(index, value))
.css(map)

jQuery.cssHooks *Obj*

Hook directly into jQuery to override how particular CSS properties are retrieved or set, normalize CSS property naming, or create custom properties.

.hasClass(className) *Str*

Determine whether any of the matched elements are assigned the given class.

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

.height() — *Int*

Get the current computed height for the first element in the set of matched elements.

.height(value)
.height(function(index, height))

.innerHeight() — *Int*

Get the current computed height for the first element in the set of matched elements, including padding but not border.

.innerWidth() — *Int*

Get the current computed width for the first element in the set of matched elements, including padding but not border.

.offset() — *Int*

Get the current coordinates of the first element in the set of matched elements, relative to the document.

.offset(coordinates) - .offset(function(index, coords))

.outerHeight([includeMargin]) — *Int*

Get the current computed height for the first element in the set of matched elements, including padding, border, and optionally margin. Returns an integer (without "px") representation of the value or null if called on an empty set of elements.

.outerWidth([includeMargin]) — *Int*

Get the current computed width for the first element in the set of matched elements, including padding and border.

.position() — *Obj*

Get the current coordinates of the first element in the set of matched elements, relative to the offset parent.

.removeClass() — *jQ*

Remove a single class, multiple classes, or all classes from each element in the set of matched elements.

.removeClass([className])
.removeClass(function(index, class))

.scrollLeft() — *Int*

Get the current horizontal position of the scroll bar for the first element in the set of matched elements.

.scrollLeft(value)

.scrollTop() — *Int*

Get the current vertical position of the scroll bar for the first element in the set of matched elements.

.scrollTop(value)

.toggleClass() — *jQ*

Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the switch argument.

.toggleClass(className)
.toggleClass(className, switch)

.width() — *Int*

Get the current computed width for the first element in the set of matched elements.

.width(value) - .width(function(index, width))

✻ EVENTS

.bind() — *jQ*

Attach a handler to an event for the elements.

.bind(eventType [, eventData], handler(eventObject))
.bind(eventType [, eventData], preventBubble)
.bind(events)

.blur() — *jQ*

Bind an event handler to the "blur" JavaScript event, or trigger that event on an element.

.blur(handler(eventObject))
.blur([eventData], handler(eventObject))

.change() — *jQ*

Bind an event handler to the "change" JavaScript event, or trigger that event on an element.

.change(handler(eventObject))
.change([eventData], handler(eventObject))

.click() — *jQ*

Bind an event handler to the "click" JavaScript event, or trigger that event on an element.

.click(handler(eventObject))
.click([eventData], handler(eventObject))

.dblclick() — *jQ*

Bind an event handler to the "dblclick" JavaScript event, or trigger that event on an element.

.dblclick(handler(eventObject))
.dblclick([eventData], handler(eventObject))

.delegate() — *jQ*

Attach a handler to one or more events for all elements that match the selector, now or in the future, based on a specific set of root elements.

.delegate(selector, eventType, handler)
.delegate(selector, eventType, eventData, handler)
.delegate(selector, events)

.die() — *jQ*

Remove all event handlers previously attached using .live() from the elements.

.die(eventType [, handler])
.die(eventTypes)

.error() — *jQ*

Bind an event handler to the "error" JavaScript event.

.error(handler(eventObject))
.error([eventData], handler(eventObject))

event.currentTarget — *jQ*

The current DOM element within the event bubbling phase.

event.data — *Any*

The optional data passed to jQuery.fn.bind when the current executing handler was bound.

★ event.delegateTarget — *el*

The element where the currently-called jQuery event handler was attached.

event.isDefaultPrevented() — *0-1*

Returns whether event.preventDefault() was ever called on this event object.

event.isImmediatePropagationStopped() — *0-1*

Returns whether event.stopImmediatePropagation() was ever called on this event object.

event.isPropagationStopped() — *0-1*

Returns whether event.stopPropagation() was ever called on this event object.

event.namespace — *Str*

The namespace specified when the event was triggered.

event.pageX — *N*

The mouse position relative to the left edge of the document.

event.pageY — *N*

The mouse position relative to the top edge of the document.

event.preventDefault() — *u*

If this method is called, the default action of the event will not be triggered.

event.relatedTarget — *el*

The other DOM element involved in the event, if any.

event.result — *Obj*

The last value returned by an event handler that was triggered by this event, unless the value was undefined.

event.stopImmediatePropagation()

Prevents other event handlers from being called.

event.stopPropagation()

Prevents the event from bubbling up the DOM tree, preventing any parent handlers from being notified of the event.

event.target — *el*

The DOM element that initiated the event.

event.timeStamp — *N*

The difference in milliseconds between the time the browser created the event and January 1, 1970.

event.type — *Str*

Describes the nature of the event.

event.which — *N*

For key or button events, this attribute indicates the specific button or key that was pressed.

.focus() — *jQ*

Bind an event handler to the "focus" JavaScript event, or trigger that event on an element

.focus(handler(eventObject))
.focus([eventData], handler(eventObject))

.focusin() — *jQ*

Bind an event handler to the "focusin" event.

.focusin(handler(eventObject))
.focusin([eventData], handler(eventObject))

.focusout() — *jQ*

Bind an event handler to the "focusout" JavaScript event.

.focusout(handler(eventObject))
.focusout([eventData], handler(eventObject))

.hover() — *jQ*

Bind two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.

.hover(handlerIn(eventObject)
.handlerOut(eventObject))
.hover(handlerInOut(eventObject))

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

.keydown() *jQ*

Bind an event handler to the "keydown" JavaScript event, or trigger that event on an element.

.keydown(handler(eventObject))
.keydown([eventData], handler(eventObject))

.keypress() *jQ*

Bind an event handler to the "keypress" JavaScript event, or trigger that event on an element.

.keypress(handler(eventObject))
.keypress([eventData], handler(eventObject))

.keyup() *jQ*

Bind an event handler to the "keyup" JavaScript event, or trigger that event on an element.

.keyup(handler(eventObject))
.keyup([eventData], handler(eventObject))

.live() *jQ*

Attach an event handler for all elements which match the current selector, now and in the future.

.live(events, handler) - .live(events, data, handler)
.live(events-map)

.load() *jQ*

Bind an event handler to the "load" JavaScript event.

.load(handler(eventObject))
.load([eventData], handler(eventObject))

.mousedown() *jQ*

Bind an event handler to the "mousedown" JavaScript event, or trigger that event on an element.

.mousedown(handler(eventObject))
.mousedown([eventData], handler(eventObject))

.mouseenter() *jQ*

Bind an event handler to be fired when the mouse enters an element, or trigger that handler on an element.

.mouseenter(handler(eventObject))
.mouseenter([eventData], handler(eventObject))

.mouseleave() *jQ*

Bind an event handler to be fired when the mouse leaves an element, or trigger that handler on an element.

.mouseleave(handler(eventObject)
.mouseleave([eventData], handler(eventObject))

.mousemove() *jQ*

Bind an event handler to the "mousemove" JavaScript event, or trigger that event on an element.

.mousemove(handler(eventObject))
.mousemove([eventData], handler(eventObject))

.mouseout() *jQ*

Bind an event handler to the "mouseout" JavaScript event, or trigger that event on an element.

.mouseout(handler(eventObject))
.mouseout([eventData], handler(eventObject))

.mouseover() *jQ*

Get the current computed width for the first element in the set of matched elements.

.mouseover(handler(eventObject))
.mouseover([eventData], handler(eventObject))

.mouseup() *jQ*

Bind an event handler to the "mouseup" JavaScript event, or trigger that event on an element.

.mouseup(handler(eventObject))
.mouseup([eventData], handler(eventObject))

★ .off() *jQ*

Remove an event handler.

.off(events [, selector] [, handler])
.off(events-map [, selector])

★ .on() *jQ*

Attach an event handler function for one or more events to the selected elements.

.on(events [, selector] [, data], handler)
.on(events-map [, selector] [, data])

.one() *jQ*

Attach a handler to an event for the elements. The handler is executed at most once per element.

.one(events [, data], handler)
.one(events [, selector] [, data], handler)
.one(events-map [, selector] [, data])

jQuery.proxy() *jQ*

Takes a function and returns a new one that will always have a particular context.

jQuery.proxy(function, context)
jQuery.proxy(context, name)

.ready(handler) *jQ*

Specify a function to execute when the DOM is fully loaded.

.resize() *jQ*

Bind an event handler to the "resize" JavaScript event, or trigger that event on an element.

.resize(handler(eventObject))
.resize([eventData], handler(eventObject))

.scroll() *jQ*

Bind an event handler to the "scroll" JavaScript event, or trigger that event on an element.

.scroll(handler(eventObject))
.scroll([eventData], handler(eventObject))

.select() *jQ*

Bind an event handler to the "select" JavaScript event, or trigger that event on an element.

.select(handler(eventObject))
.select([eventData], handler(eventObject))

.submit() *jQ*

Bind an event handler to the "submit" JavaScript event, or trigger that event on an element.

.submit(handler(eventObject))
.submit([eventData], handler(eventObject))

.toggle() *jQ*

Bind two or more handlers to the matched elements, to be executed on alternate clicks.

handler(eventObject), handler(eventObject) [, handler(eventObject)]

.trigger() *jQ*

Execute all handlers and behaviors attached to the matched elements for the given event type.

.trigger(eventType, extraParameters)
.trigger(event)

.triggerHandler() *jQ*

Execute all handlers attached to an element for an event.

.triggerHandler(eventType, extraParameters)

.unbind() *jQ*

Remove a previously-attached event handler from the elements.

.unbind([eventType] [, handler(eventObject)])
.unbind(eventType, false)

.undelegate() *jQ*

Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

.undelegate(selector, eventType)
.undelegate(selector, eventType, handler)
.undelegate(selector, events)
.undelegate(namespace)

.undelegate() *jQ*

Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.

.undelegate(selector, eventType)
.undelegate(selector, eventType, handler)
.undelegate(selector, events)
.undelegate(namespace)

.unload() *jQ*

Bind an event handler to the "unload" JavaScript event.

.unload(handler(eventObject))
.unload([eventData], handler(eventObject))

✻ EFFECTS

.animate() *jQ*

Perform a custom animation of a set of CSS properties.

.animate(properties [, duration] [, easing] [, complete])
.animate(properties, options)

.clearQueue([queueName]) *jQ*

Remove from the queue all items that have not yet been run.

.delay(duration [, queueName]) *jQ*

Set a timer to delay execution of subsequent items in the queue.

.dequeue([queueName]) *jQ*

Execute the next function on the queue for the matched elements.

.fadeIn() *jQ*

Display the matched elements by fading them to opaque.

.fadeIn([duration] [, callback])
.fadeIn([duration] [, easing] [, callback])

.fadeOut() *jQ*

Hide the matched elements by fading them to transparent.

.fadeOut([duration] [, callback])
.fadeOut([duration] [, easing] [, callback])

.fadeTo() *jQ*

Adjust the opacity of the matched elements.

.fadeTo(duration, opacity [, callback])
.fadeTo(duration, opacity [, easing] [, callback])

.fadeToggle([duration] [, easing] [, callback]) *jQ*

Display or hide the matched elements by animating their opacity.

jQUERY 1.7 VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

jQuery.fx.interval — *N*

The rate (in milliseconds) at which animations fire.

jQuery.fx.off — *0-1*

Globally disable all animations.

.hide() — *jQ*

Hide the matched elements.

.hide(duration [, callback])
.hide([duration] [, easing] [, callback])

.queue() — *a()* *jQ*

Show the queue of functions to be executed on the matched elements.

.queue([queueName])
.queue([queueName] , newQueue)
.queue([queueName] , callback(next))

.show() — *jQ*

Display the matched elements.

.show(duration [, callback])
.show([duration] [, easing] [, callback])

.slideDown() — *jQ*

Display the matched elements with a sliding motion.

.slideDown([duration] [, callback])
.slideDown([duration] [, easing] [, callback])

.slideToggle() — *jQ*

Display or hide the matched elements with a sliding motion.

.slideToggle([duration] [, callback])
.slideToggle([duration] [, easing] [, callback])

.slideUp() — *jQ*

Hide the matched elements with a sliding motion.

.slideUp([duration] [, callback])
.slideUp([duration] [, easing] [, callback])

★ .stop() — *jQ*

Stop the currently-running animation on the matched elements.

.stop([clearQueue] [, jumpToEnd])
.stop([queue] [, clearQueue] [, jumpToEnd])

.toggle() — *jQ*

Display or hide the matched elements.

.toggle([duration] [, callback])
.toggle([duration] [, easing] [, callback])
.toggle(showOrHide)

✻ AJAX

jQuery.ajax() — *jqXHR*

Perform an asynchronous HTTP (Ajax) request.

jQuery.ajax(url [, settings])
jQuery.ajax(settings)

.ajaxComplete() — *jQ*

Register a handler to be called when Ajax requests complete. This is an Ajax Event.

.ajaxComplete(handler(event, XMLHttpRequest, ajaxOptions))

.ajaxError() — *jQ*

Register a handler to be called when Ajax requests complete with an error. This is an Ajax Event.

.ajaxError(handler(event, jqXHR, ajaxSettings, thrownError))

jQuery.ajaxPrefilter() — *u*

Handle custom Ajax options or modify existing options before each request is sent and before they are processed by $.ajax().

jQuery.ajaxPrefilter([dataTypes] , handler(options, originalOptions, jqXHR))

.ajaxSend() — *jQ*

Attach a function to be executed before an Ajax request is sent. This is an Ajax Event.

.ajaxSend(handler(event, jqXHR, ajaxOptions))

jQuery.ajaxSetup(options)

Set default values for future Ajax requests.

.ajaxStart(handler()) — *jQ*

Register a handler to be called when the first Ajax request begins. This is an Ajax Event.

.ajaxStop(handler()) — *jQ*

Register a handler to be called when all Ajax requests have completed. This is an Ajax Event.

.ajaxSuccess() — *jQ*

Attach a function to be executed whenever an Ajax request completes successfully. This is an Ajax Event.

.ajaxSuccess(handler(event, XMLHttpRequest, ajaxOptions))

jQuery.get() — *jqXHR*

Load data from the server using a HTTP GET request.

jQuery.get(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

jQuery.getJSON() — *jqXHR*

Load JSON-encoded data from the server using a GET HTTP request.

jQuery.getJSON(url [, data] [, success(data, textStatus, jqXHR)])

jQuery.getScript() — *jqXHR*

Load a JavaScript file from the server using a GET HTTP request, then execute it.

jQuery.getScript(url [, success(data, textStatus)])

.load() — *jqXHR*

Load data from the server and place the returned HTML into the matched element.

.load(url [, data] [, complete(responseText, textStatus, XMLHttpRequest)])

jQuery.param() — *Str*

Create a serialized representation of an array or object, suitable for use in a URL query string or Ajax request.

jQuery.param(obj)
jQuery.param(obj, traditional)

jQuery.post() — *jqXHR*

Load data from the server using a HTTP POST request.

jQuery.post(url [, data] [, success(data, textStatus, jqXHR)] [, dataType])

.serialize() — *Str*

Encode a set of form elements as a string for submission.

.serializeArray() — *a()*

Encode a set of form elements as an array of names and values.

✻ UTILITIES

jQuery.boxModel — *0-1*

Deprecated

jQuery.browser — *Map*

Deprecated

.clearQueue([queueName]) — *jQ*

Remove from the queue all items that have not yet been run.

jQuery.contains(container, contained) — *0-1*

Check to see if a DOM element is within another DOM element.

jQuery.data() — *Obj*

Store arbitrary data associated with the specified element. Returns the value that was set.

jQuery.data(element , key , value)
jQuery.data(element, key)
jQuery.data(element)

.dequeue([queueName]) — *jQ*

Execute the next function on the queue for the matched elements.

jQuery.each() — *Obj*

A generic iterator function, which can be used to seamlessly iterate over both objects and arrays.

jQuery.each(collection, callback(indexInArray, valueOfElement))

jQuery.extend() — *Obj*

Merge the contents of two or more objects together into the first object.

jQuery.extend(target [, object1] [, objectN])
jQuery.extend([deep], target, object1 [, objectN])

jQuery.globalEval(code)

Execute some JavaScript code globally.

jQuery.each(collection, callback(indexInArray, valueOfElement))

jQuery.grep() — *a()*

Finds the elements of an array which satisfy a filter function. The original array is not affected.

jQuery.grep(array, function(elementOfArray, indexInArray) [, invert])

jQuery.inArray() — *N*

Search for a specified value within an array and return its index (or -1 if not found).

jQuery.inArray(value, array [, fromIndex])

jQuery.isArray(obj) — *0-1*

Determine whether the argument is an array.

jQuery.isEmptyObject(object) — *0-1*

Check to see if an object is empty (contains no properties).

jQuery.isFunction(object) — *0-1*

Determine if the argument passed is a Javascript function object.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

★ jQuery.isNumeric(object) — *0-1*

Determines whether its argument is a number.

jQuery.isPlainObject(object) — *0-1*

Check to see if an object is a plain object (created using "{}" or "new Object").

jQuery.isWindow(object) — *0-1*

Determine whether the argument is a window.

jQuery.isXMLDoc(object) — *0-1*

Check to see if a DOM node is within an XML document (or is an XML document).

jQuery.makeArray(object) — *a()*

Convert an array-like object into a true JavaScript array.

jQuery.map() — *a()*

Translate all items in an array or object to new array of items.

jQuery.map(array, callback(elementOfArray, indexInArray))
jQuery.map(arrayOrObject, callback(value, indexOrKey))

jQuery.merge(first, second) — *a()*

Merge the contents of two arrays together into the first array.

jQuery.noop() — *f(x)*

An empty function.

jQuery.now() — *N*

Return a number representing the current time.

jQuery.parseJSON(json) — *Obj*

Takes a well-formed JSON string and returns the resulting JavaScript object.

jQuery.parseXML(data) — *XMLdoc*

Parses a string into an XML document.

jQuery.proxy() — *f(x)*

Takes a function and returns a new one that will always have a particular context.

jQuery.proxy(function, context)
jQuery.proxy(context, name)

jQuery.queue() — *a()* *jQ*

Takes a function and returns a new one that will always have a particular context.

jQuery.queue(element [, queueName])
jQuery.queue(element , queueName , newQueue)

.queue() — *a()* *jQ*

Show the queue of functions to be executed on the matched elements.

.queue([queueName])
.queue([queueName] , newQueue)
.queue([queueName] , callback(next))

jQuery.removeData() — *jQ*

Remove a previously-stored piece of data

jQuery.removeData(element [, name])

jQuery.support — *obj*

A collection of properties that represent the presence of different browser features or bugs.

jQuery.trim(str) — *Str*

Remove the whitespace from the beginning and end of a string.

jQuery.type(obj) — *Str*

Determine the internal JavaScript [[Class]] of an object.

jQuery.unique(array) — *a()*

Sorts an array of DOM elements, in place, with the duplicates removed. Note that this only works on arrays of DOM elements, not strings or numbers.

✻ CALLBACKS OBJECT

★ jQuery.Callbacks(flag)

A multi-purpose callbacks list object that provides a powerful way to manage callback lists.

★ callbacks.add(callbacks) — *u*

Add a callback or a collection of callbacks to a callback list.

★ callbacks.disable() — *u*

Disable a callback list from doing anything more.

★ callbacks.empty() — *u*

Remove all of the callbacks from a list.

★ callbacks.fire(arguments) — *u*

Call all of the callbacks with the given arguments

★ callbacks.fired() — *0-1*

Determine if the callbacks have already been called at least once.

★ callbacks.fireWith() — *u*

Call all callbacks in a list with the given context and arguments.

callbacks.fireWith([context] [, args])

★ callbacks.has(callback) — *0-1*

Determine whether a supplied callback is in a list.

★ callbacks.lock() — *u*

Lock a callback list in its current state.

★ callbacks.locked() — *0-1*

Determine if the callbacks list has been locked.

★ callbacks.remove(callback) — *u*

Remove a callback or a collection of callbacks from a callback list.

✻ DATA & MISCELLANEOUS

.data() — *jQ*

Store arbitrary data associated with the matched elements.

data(key , value), data(key)

jQuery.dequeue() — *jQ*

Execute the next function on the queue for the matched element.

jQuery.dequeue(element [, queueName])

jQuery.hasData(element) — *0-1*

Determine whether an element has any jQuery data associated with it.

★ .removeData() — *jQ*

Remove a previously-stored piece of data.

.removeData([name]), .removeData([list])

.each() — *jQ*

Iterate over a jQuery object, executing a function for each matched element.

.each(function(index, Element))

.get([index]) — *el* *a()*

Retrieve the DOM elements matched by the jQuery object.

.each(function(index, Element))

.index — *Num*

Search for a given element from among the matched elements.

.index(selector), .index(element)

jQuery.noConflict([removeAll]) — *Obj*

Relinquish jQuery's control of the $ variable.

.size() — *Num*

Return the number of elements in the jQuery object.

.toArray() — *a()*

Retrieve all the DOM elements contained in the jQuery set, as an array.

✻ DEFERRED

deferred.always() — *def*

Add handlers to be called when the Deferred object is either resolved or rejected.

deferred.always(alwaysCallbacks [, alwaysCallbacks])

deferred.done() — *def*

Add handlers to be called when the Deferred object is resolved.

deferred.done(doneCallbacks [, doneCallbacks])

deferred.fail() — *def*

Add handlers to be called when the Deferred object is rejected.

deferred.fail(failCallbacks [, failCallbacks])

deferred.isRejected() — *0-1*

Determine whether a Deferred object has been rejected.

deferred.isResolved() — *0-1*

Determine whether a Deferred object has been resolved.

deferred.notify(args) — *0-1*

Call the progressCallbacks on a Deferred object with the given args.

jQUERY 1.7
VISUAL CHEAT SHEET

★ = NEW OR CHANGED IN jQUERY 1.7 / *f(x)* = FUNCTION / *a* = ARRAY / *jQ* = jQUERY / *El* = ELEMENT / 0-1 = BOOLEAN / *Obj* = OBJECT / *NUM* = NUMBER / *Str* = STRING

deferred.notifyWith() *def*

Call the progressCallbacks on a Deferred object with the given context and args.

deferred.notifyWith(context [, args])

deferred.pipe() *promise*

Utility method to filter and/or chain Deferreds.

deferred.pipe([doneFilter] [, failFilter])
deferred.pipe([doneFilter] [, failFilter] [, progressFilter])

deferred.progress() *def*

Add handlers to be called when the Deferred object generates progress notifications.

deferred.progress(progressCallbacks)

deferred.promise() *promise*

Return a Deferred's Promise object.

deferred.promise([target])

deferred.reject(**args**) *def*

Reject a Deferred object and call any failCallbacks with the given args.

deferred.rejectWith() *def*

Reject a Deferred object and call any failCallbacks with the given context and args.

deferred.rejectWith(context [, args])

deferred.resolve(**args**) *def*

Resolve a Deferred object and call any doneCallbacks with the given args.

deferred.resolveWith() *def*

Resolve a Deferred object and call any doneCallbacks with the given args.

deferred.resolveWith(context [, args])

deferred.state() *Str*

Determine the current state of a Deferred object.

deferred.then() *def*

Add handlers to be called when the Deferred object is resolved or rejected.

deferred.then(doneCallbacks, failCallbacks)
deferred.then(doneCallbacks, failCallbacks [, progressCallbacks])

.promise() *promise*

Return a Promise object to observe when all actions of a certain type bound to the collection, queued or not, have finished.

.promise([type] [, target])

jQuery.when(**deferreds**) *promise*

Provides a way to execute callback functions based on one or more objects, usually Deferred objects that represent asynchronous events.

New or Changed in jQuery 1.7.1

WHAT'S NEW?

Aspects of the API that were changed in the corresponding version of jQuery. API changes in jQuery 1.7.0 dealt primarily with the new Event APIs: .on() and .off()

Better Support for HTML5 in IE6/7/8

jQuery.Callbacks()

Toggling Animations Work Intuitively

jQuery.Callbacks()
callbacks.add()
callbacks.disable()
callbacks.empty()
callbacks.fire()
callbacks.fired()
callbacks.fireWith()
callbacks.has()
callbacks.lock()
callbacks.locked()
callbacks.remove()
deferred.notify()
deferred.notifyWith()
deferred.pipe()
deferred.progress()
deferred.state()
deferred.then()
event.delegateTarget
.is()
jQuery.isNumeric()
.off()
.on()
.removeAttr()
.removeData()
.stop()

About jQuery

THE WRITE LESS, DO MORE, JAVASCRIPT LIBRARY

jQuery is a fast and concise JavaScript Library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is designed to change the way that you write JavaScript.

jQuery Official Web Site http://jquery.com
Download http://docs.jquery.com/Downloading_jQuery
Documentation http://docs.jquery.com/Main_Page
Tutorials http://docs.jquery.com/Tutorials
Bug Traker http://bugs.jquery.com/newticket
Discussion http://docs.jquery.com/Discussion

Credits

JQUERY 1.7 VISUAL CHEAT SHEET

This jQuery visual cheat sheet is designed by Antonio Lupetti and is distributed for free.

Update as of december 2011, jQuery current release v 1.7.1

My Blog http://woorkup.com
Twitter @woork - http://twitter.com/woork
Facebook http://facebook.com/antoniolupetti
Google+ https://plus.google.com/u/0/112930363515083491757

Om indholdet

Denne PDF-fil indeholder et udkast til et kapitel som måske vil blive inkluderet i en eventuel næste udgave af ”Bogen om ASP.NET 4.0 Web Forms” skrevet af Michell Cronberg. Den første udgave af bogen blev udgivet omkring sommerferien 2010, og bogen kan købes hos boghandlere og på nettet (prøv eventuelt bogpriser.dk – søg efter ISBN 978-87-993382-0-7). Se også http://www.bogenomaspnet.dk.

Filen kan frit distribueres under forudsætning af, at indholdet ikke ændres.

Fejl, mangler og kommentarer modtages gerne (se blog.cronberg.dk for kontaktoplysninger).

God fornøjelse,

Odense, november 2010
Michell Cronberg
http://blog.cronberg.dk

Introduktion til jQuery

Når man udvikler en ASP.NET-applikation skal man tage stilling til, om man ønsker at skabe en applikation med en traditionel klient/server-orienteret HTML-brugerflade, eller en mere dynamisk, hurtig og tiltalende brugerflade, hvor man holder så meget funktionalitet som muligt på klienten (browseren). Ønsker man den sidstnævnte type (som anvendes i mange webapplikationer i dag) er man tvunget til at bruge CSS (se side 26) og ikke mindst JavaScript (se side 58). Fordi standard JavaScript i realiteten er rimelig "rå" i sin måde at udvælge og manipulere elementer på en HTML-side, og samtidig ikke nødvendigvis fungerer på samme måde i forskellige typer og versioner af browsere, er det faktisk ret kompliceret at skabe en dynamisk klientbaseret brugerflade. Derfor er der i løbet af de seneste år dukket en del forskellige JavaScript-biblioteker op, som gør det nemmere at bruge JavaScript. Dels slipper man for at skrive så meget kode, fordi man kan benytte funktionalitet, som allerede er kodet i bibliotekerne, og dels behøver man ikke i samme grad være bevidst om, hvorvidt koden nu også virker i alle browsere.

jQuery

Et af de mest populære biblioteker hedder jQuery, og er primært skabt af JavaScript-guruen John Resig, men der er en masse andre, som har hjulpet med udvikling og vedligeholdelse af projektet. De fremgår alle af jQuery-sitet under "About", hvor også historien bag jQuery er beskrevet.

jQuery har primært til formål at gøre det nemmere at skrive effektiv, hurtig og browseruafhængig JavaScript til at udvælge HTML-elementer og herefter manipulere disse på mange forskellige måder. Samtidig består selve jQuery-kernen af forskellige metoder til eksempelvis at gennemløbe elementer, foretage AJAX-kald og meget mere. Udover selve kernen er der et par yderligere biblioteker, man kan vælge at benytte – herunder et bibliotek med konkrete brugerfladeelementer som eksempelvis en "kontrol" til at vælge dato og tid, en komponent til at sortere lister, en komponent til at implementere drag-and-drop og meget mere. I skrivende stund (ultimo 2010)

findes der ligeledes en betaversion af et bibliotek, der kan bruges til webapplikationer til afvikling på smartphones (iPhone, Android med videre).

jQuery-biblioteket er umådelig populært og benyttes af rigtig mange store websites i hele verden (Google, Dell, CBS, WordPress med videre).

jQuery og Microsoft

jQuery har som udgangspunkt intet med Microsoft at gøre, og er dermed heller ikke direkte relateret til ASP.NET. jQuery manipulerer og tilføjer HTML-elementer i browseren ved hjælp af JavaScript, og det er ligegyldigt, hvordan HTML-opmærkningerne er skabt. De kan komme fra serverorienterede teknologier som eksempelvis ASP.NET eller PHP, men det kan også være fra statiske HTML-sider.

jQuery er et Open Source-projekt, der hviler i sig selv, men Microsoft har for nogle år siden fået øjnene op for, hvor effektive bibliotekerne er, og har fået lov til at distribuere projektet sammen med Visual Studio. Samtidig er Visual Studio tilrettet, så der er Intellisense til de forskellige biblioteker. Med tiden vil Microsoft levere indhold til jQuery-projektet som tak, og jQuery vil blive tættere integreret i ASP.NET i eksempelvis ASP.NET AJAX (se side 171) og AJAX Control Toolkit (se side 198).

Et kort eksempel på brug af jQuery

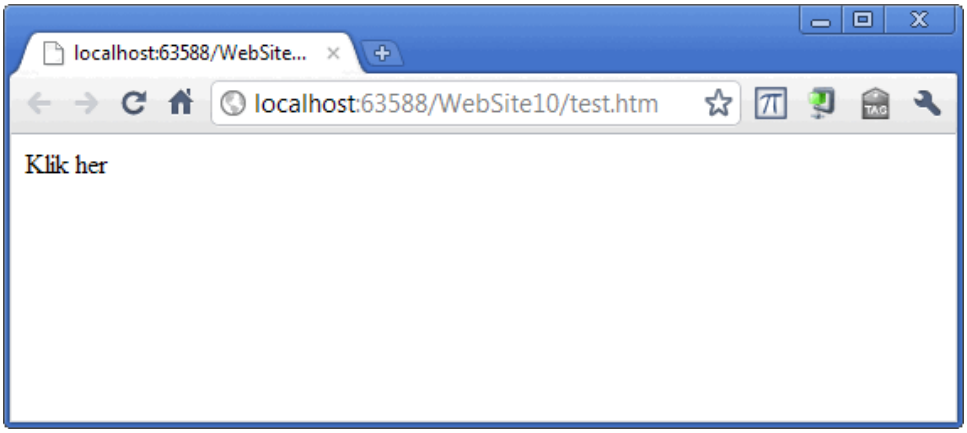
Inden vi ser nærmere på jQuery, kan det være på sin plads med et kort eksempel på, hvad jQuery kan.

Prøv at tilføje en almindelige HTML-side til et website i Visual Studio med følgende indhold:

```
<html>
<head>
 <script
  src="http://ajax.microsoft.com/ajax/jquery/jquery-1.4.2.min.js"
  type="text/javascript">
 </script>
 <style type="text/css">
```

```
   .demo
   {
    background-color: Yellow;
    font: normal bold 26px Arial;
    border: solid 1px black;
    text-align: center;
    padding: 10px;
   }
  </style>
</head>
<body>
  <p style="display: none" id="tekst">
   Dette er en test af jQuery
  </p>
  <p id="klik">
   Klik her
  </p>
  <script type="text/javascript">
   $("#klik").click(function () {
    $("#tekst").addClass("demo").show("slow");
    $("#klik").hide();
   });
  </script>
</body>
</html>
```

Når siden vises i en browser (højreklik på filen i Solution Explorer og vælg ”View in browser”) vises blot en side med teksten ”Klik her”. Her er eksempelvis siden vist i Chrome:



Figur 214. jQuery demo (1).

Når der klikkes et sted i ”Klik her” vil teksten ”Dette er en test af jQuery” blive vist animeret (teksten flyver ind fra toppen af siden) tilføjet noget CSS, og ”Klik her” fjernes. Den animerede del er logisk nok svær at vise på et billede – men her er det endelige resultat.

Figur 215. jQuery demo (2).

Eksemplet er meget simpelt, men viser hvor lidt JavaScript (tre linjer) der skal til at kode en ellers ret kompliceret funktionalitet. Det viser samtidig fire af de elementer som gennemgås i resten af kapitlet:

- Hvordan man *inkluderer* jQuery på en side (i eksemplet hentes biblioteket fra en af Microsofts servere).
- Hvordan man kan *finde* elementer på en side (eksempelvis findes paragraffen kaldet klik ved hjælp af $(”#klik”)).
- Hvordan man nemt kan *binde* script op til en hændelse (når der klikkes på teksten så …).

- Hvordan man kan tilføje CSS til et element og benytte en animeret *effekt* (show-metoden).

I det følgende gennemgås grundlæggende brug af jQuery, men det anbefales at se på de mange ressourcer på http://jquery.com.

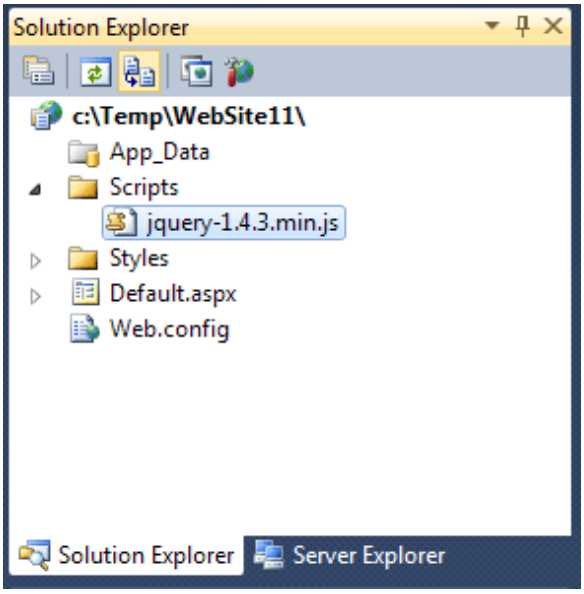
Inkludering af jQuery

For at kunne benytte jQuery skal man som minimum tilføje selve jQuery-kernebiblioteket til den side, hvor det skal benyttes. Filen findes i forskellige versioner:

- jquery-x.x.x.js er den fulde version (også kaldet source). Den er "læsbar" i en editor og benyttes i forbindelse med udvikling/debug.
- jquery-x.x.x.min.js er den komprimerede version (også kaldet minified) og benyttes i produktion. Her er eksempelvis alle linjeskift og whitespaces fjernet (hvilket gør den komplet umulig at læse i en editor – men den fylder til gengæld ikke så meget).
- jquery-x.x.x-vsdoc.js er den fil, man skal bruge, hvis man ønsker Intellisense i Visual Studio. Den bør (ligesom source-versionen) kun benyttes i forbindelse med udvikling.

I de nævnte filnavne repræsenterer x.x.x versionsnummeret som i skrivende stund hedder 1.4.3.

Man kan vælge at hente den sidste nye version direkte fra jQuery og så placere filen i en valgfri lokal mappe – eksempelvis /scripts.



Figur 216. jQuery-fil placeret i en mappe – klar til brug.

Filen kan nu trækkes ind i head-opmærkningen på en side:

```
<html>
<head runat="server">
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
</head>
<!-body og resten af siden fjernet -->
```

Man kan også vælge blot at henvise til filen på nettet – enten hos jQuery, Microsoft eller Google som alle har scripts liggende tilgængelige på en URL. Det har den fordel, at man ikke behøver bøvle med at have filen liggende lokalt, og ikke mindst at man kan opnå store cache-fordele ved at benytte en .js-fil fra en URL, som man måske allerede har hentet tidligere, og som dermed er cachet i browseren. Man kan benytte en af følgende adresser (her version 1.4.3):

- http://code.jquery.com/jquery-1.4.3.min.js
- http://ajax.googleapis.com/ajax/libs/jquery/1.4.3/jquery.min.js
- http://ajax.microsoft.com/ajax/jquery/jquery-1.4.3.min.js

Her bruges scriptet fra jQuery i stedet for et lokalt script.

```
<html>
<head runat="server">
```

```
 <script src="http://code.jquery.com/jquery-1.4.3.min.js"
  type="text/javascript"></script>
</head>
<!-body og resten af siden fjernet -->
```

Om man bruger et lokalt script eller et script fra nettet (scriptet leveres gennem et såkaldt CDN – Content Delivery Network) kommer an på, hvilken applikation man bygger. Hvis der er tale om et intranet, er et lokalt script nok at foretrække, men hvis scriptet skal bruges i et offentligt site med mange brugere, er et script fra Google eller Microsoft at foretrække.

Hvis man udvikler et site med en master page, kan man vælge at tilføje scriptet til den, men det har den ulempe, at alle sider så vil få scriptet inkluderet. Hvis man ikke bruger jQuery på alle sider (og det gør man ikke nødvendigvis), så er det spild af ressourcer. Alternativt kan man tilføje scriptet til en Content-kontrol (se side 250) der består af en ContentPlaceHolder-kontrol placeret i head-opmærkningen:

```
<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
</asp:Content>
```

Grundlæggende jQuery

Når jQuery er loadet i browseren, kan man tilgå de forskellige objekter. Det primære objekt kaldes logisk nok jQuery og kan eksempelvis tilgås som:

```
jQuery("#minId").text("Hej fra jQuery");
```

Man vil dog tit bruge et dollartegn ($) som genvej til objektet. Derfor kan førnævnte ligeså godt skrives som:

```
$("#minId").text("Hej fra jQuery");
```

I eksemplerne rundt på nettet benytter man mest $-genvejen.

Som det gennemgås senere, giver jQuery tilgang til masse hændelser, men en af dem benyttes en del og gennemgås derfor kort her. Den hedder ready og afvikles, når

browseren er færdig med at skabe objekterne på siden (DOM). Hændelsen bruges derfor til kode, som man ønsker afviklet, når siden er loadet og ”klar”. Man vil typisk se hændelsen bundet til en metode på en af følgende måder:

- $(document).ready(handler)
- $(handler)

Det giver samme resultat – nemlig at funktionen (i ovennævnte eksempel kaldet handler) kaldes, når DOM er klar. Hvis førnævnte kode skal kaldes, når siden er loadet, kan man altså eksempelvis gøre som følger:

```
<script type="text/javascript">
 $(document).ready(handler);

 function handler() {
  $("#minId").text("Hej fra jQuery");
 }
</script>
```

Man vil dog typisk benytte en såkaldt anonym metode, der defineres og benyttes i stedet for en reference til en metode (metoden bag en hændelse bliver sjældent brugt til andet alligevel, og man kan lige så godt trække så meget performance ud af JavaScript som muligt). En ready-hændelse (og andre hændelser) kodes derfor tit som:

```
<script type="text/javascript">

 $(document).ready(function () {
  $("#minId").text("Hej fra jQuery");
 });

</script>
```

Man behøver ikke benytte en anonym metode, men en stor del af eksemplerne på nettet vil bruge dem, og man kan derfor lige så godt vende sig til syntaksen. Den kan være lidt svær at få fat på, men hvis man sørger for at skrive hele skabelonen for en anonym metode, er det lidt nemmere – eksempelvis:

```
$(”#minId”).click(function () {} );
```

Så kan man efterfølgende placere et linjeskift imellem de to tuborgklammer og skrive sin kode.

Brug af selectors

Noget af det jQuery er rigtig god til er at finde HTML-elementer på en side, som efterfølgende kan påvirkes på forskellig vis. Elementerne findes ved hjælp af de såkaldte selectors, og syntaksen bag en selector minder en del om måden CSS finder elementer på. jQuery har dog udvidet denne funktionalitet en hel del.

Her gennemgås kun de mest simple selectors, og det anbefales at se på dokumentationen på jquery.com for yderligere information.

En selector ser ud som følger:

```
$("[selector]")
```

hvor [selector] er et udtryk, som kan antage mange former. Her er nogle simple eksempler:

Selector	Forklaring
$("*")	Vælger alle elementer
$("a")	Vælger alle a-opmærkninger (link)
$("li")	Vælger alle li-opmærkninger (elementer på en liste)
$("span, div")	Vælger alle span- og div-elementer
$("p a")	Vælger alle links, men kun i afsnit
$("ul li")	Vælger alle li (elementer) i en ul (og ikke eksempelvis i ol)
$(":button")	Vælger alle knapper
$(":checkbox")	Vælger alle checkbokse

Selector	Forklaring
$("#mitId")	Vælger elementet med mitId som ID
$(".minKlasse")	Vælger elementer med (CSS-) klassen minKlasse
$("#mitId .minKlasse")	Vælger elementer der har klassen minKlasse påført, og som kan findes i elementet med ID mitId

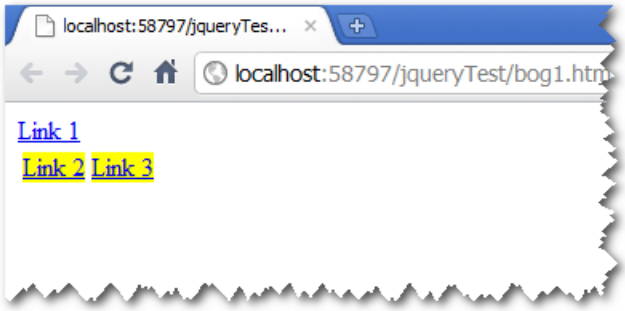
Tabel 29. Simple selectorer.

Som eksempel kan vises følgende:

```
<html>
<head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {
   $("#minTabel a").css("background-color", "yellow");
  });
 </script>
</head>
<body>
 <a href="#">Link 1</a>
 <table id="minTabel">
  <tr>
   <td>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
   </td>
  </tr>
 </table>
</body>
</html>
```

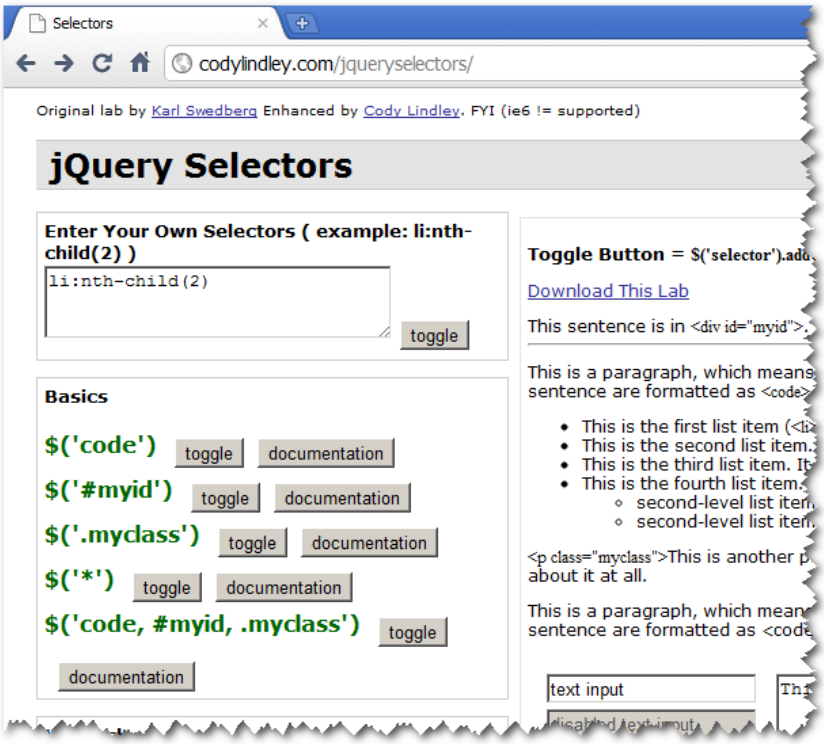
Siden består af tre link (a-opmærkninger) – et uden for en tabel og to i en tabel. Når siden er loadet (jQuery's ready-hændelse) findes de link, som er i tabellen, ved hjælp

af $("#minTabel a"), og de fundne link tilføjes en baggrundsfarve ved hjælp af CSS-metoden (herom senere). Resultat bliver følgende:



Figur 217. To link påvirket af jQuery.

Der findes en rigtig god side på http://kortlink.dk/8bkz, som giver mange ”live” eksempler på forskellige typer af selectors. Hvis man er helt ny i jQuery, bør man bruge lidt tid på at ”lege” på denne side – den giver en rigtig god forståelse for, hvordan man nemt kan finde elementer.



Figur 218. jQuery selectors læres nemmest ved at lege – prøv selv på http://kortlink.dk/8bkz.

Når først man har fundet de elementer, man skal arbejde med, ved hjælp af selectors, kan man efterfølgende påvirke elementerne på forskellig måde eller fange og benytte hændelser.

Manipulering af elementer

jQuery-objektet indeholder et hav af forskellige metoder, som kan manipulere et eller flere elementer. Her er nogle af de vigtigste:.

Metode	Forklaring
addClass()	Tilføjer en CSS-klasse
before(), after()	Tilføjer indhold før eller efter et element
attr()	Læser eller sætter en attribut
css()	Læser eller tilføjer CSS
html()	Læser eller tilføjer HTML-indhold
removeAttr()	Fjerner en attribut
removeClass()	Fjerner en CSS-klasse
val()	Finder værdien (eksempelvis teksten i en tekstboks)

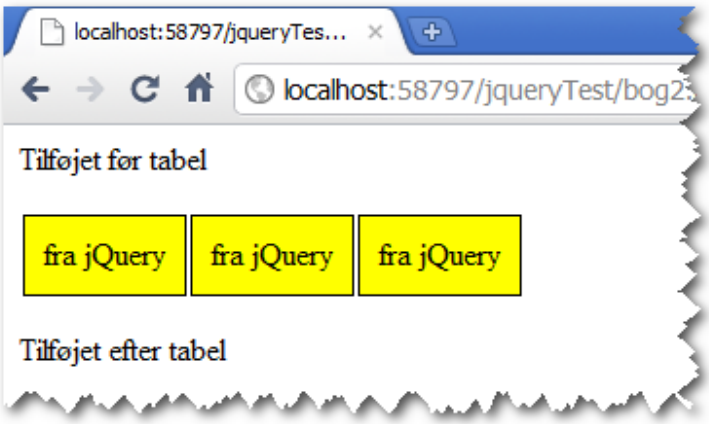
Tabel 30. Tabel over nogle få jQuery-metoder.

Her er et eksempel, som benytter nogle af metoderne på en tabel:

```
<html><head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {
   $("#minTabel").before("<p>Tilføjet før tabel</p>");
   $("#minTabel td").css("background-color", "yellow");
   $("#minTabel td").addClass("demo");
   $("#minTabel td").html("fra jQuery");
```

```
    $("#minTabel").after("<p>Tilføjet efter tabel</p>");
  });
 </script>
 <style type="text/css">
  .demo { padding: 10px; border: 1px solid black; }
 </style>
</head><body>
 <table id="minTabel">
  <tr><td /><td /><td /></tr>
 </table>
</body>
</html>
```

Siden består af en meget simpel tabel med tre celler uden indhold, men ved hjælp af jQuery tilføjes indhold til cellerne (HTML-metoden), baggrundsfarven ændres (CSS-metoden), der tilføjes en CSS-klasse (addClass-metoden), og der tilføjes sågar en paragraf før og efter tabellen (before- og after-metoderne).



Figur 219. Manipulering af elementer ved hjælp af jQuery.

Effekter

Der findes ligeledes en del metoder, som manipulerer med elementer på mere speciel måde, og flere af dem er animeret i en eller anden form. Her er et par af dem beskrevet:

Metode	Forklaring
fadeIn(), fadeOut()	Viser og skjuler et element ved hjælp af en ”fade” animation
show(), hide()	Viser og skjuler et element (eventuelt ”eksplosion” animation)
slideDown(), slideUp()	Viser og skjuler et element ved hjælp af en animation som ”trækker” elementet ned/op.

Tabel 31. Forskellige simple jQuery-effekter.

Følgende script giver en ide om, hvad metoderne kan, men prøv det selv for at se den endelige effekt:

```
<html>
<head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {
   $(".demo").delay(500).show("slow").delay(500).hide("slow");
   $(".demo").delay(500).fadeIn("slow").delay(500).fadeOut("slow");
   $(".demo").delay(500).slideDown("slow").delay(500).slideUp("slow");
  });
 </script>
 <style type="text/css">
  .demo
  {
   display: none;
   padding: 10px;
   background-color: Yellow;
   border: 1px solid black;
   width: 200px;
  }
 </style>
</head>
<body>
 <div class="demo">
```

```
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur
    consequat metus vitae est. Duis quis metus. Lorem ipsum dolor sit amet,
    consectetuer adipiscing elit. Quisque scelerisque.
  </div>
</body>
</html>
```

Siden opretter en div-opmærkning, som ved hjælp af CSS er stylet en smule og skjult, når siden loader. Når jQuery er klar til at afvikle script, benyttes de forskellige effekter, og som det kan ses, kan metoderne kaldes på forskellig måde. I scriptet bruges delay-metoden til at skabe et par pauser, så animationerne ikke kører lige efter hinanden, og koden er ligeledes et eksempel på, hvordan man i jQuery kan kæde metodekald efter hinanden.

Som tidligere nævnt skal eksemplet ses ”live” i en browser for at få en ide om, hvordan animationerne virker.

Figur 220. Hide-metoden i funktion - elementet bliver fjernet langsomt fra browseren.

Brug af hændelser

På en dynamisk brugerflade vil brugeren forvente, at siden reagerer på hændelser som eksempelvis et klik på en knap, en mouseover, en ændring i en tekstboks eller lignende. jQuery tilbyder en nem måde at binde forskellige hændelser fra elementer

på en side til noget kode. Her er et par vigtige metoder, som binder en hændelse til en metode:

Hændelse	**Forklaring**
focusin(), focusout()	Når et element modtager eller taber fokus (eksempelvis ved tabulering).
change()	Når et element ændres (eksempelvis teksten i en tekstboks).
click()	Når der klikkes på elementet (klik på en knap, klik i en paragraf med videre).
keydown(), keyup()	Når der trykkes og slippes en tast.
hover()	Når musen føres over et element.

Tabel 32. Tabel over et par metoder der binder hændelser til metoder.

Her er et kort eksempel på brugen af en click- og en hover-hændelse:

```
<html><head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {

   $(".demo").click(function () {
    $(this).slideUp("slow");
   });

   $(".demo").hover(function () {
    $(this).css("background-color", "Blue");
   }, function () {
    $(this).css("background-color", "Yellow");
   });

  });
 </script>
 <style type="text/css">
```

```
   .demo
   {
    padding: 10px;
    background-color: Yellow;
    border: 1px solid black;
    width: 200px;
   }
  </style>
</head>
<body>
  <div class="demo">
   Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur
   consequat metus vitae est. Duis quis metus. Lorem ipsum dolor sit amet,
   consectetuer adipiscing elit. Quisque scelerisque.
  </div>
</body></html>
```

Siden danner en boks med lidt tekst, og når musen føres over boksen ændres baggrundsfarven (hover), og når der klikkes et sted i boksen (click), vil den forsvinde ved hjælp af slideUp-animationen. Koden viser samtidig brugen af this-kodeordet, som kan bruges til at få fat i det element, som har resulteret i hændelsen.

AJAX

jQuery har flere muligheder for at foretage AJAX-kald (se også side 171), og i nogen situationer er det nemmere at bruge jQuery til AJAX, end de muligheder ASP.NET stiller til rådighed.

Der findes eksempelvis metoden get() som foretager en rå HTTP GET-forespørgsel, og metoden load() som foretager en GET eller POST og placerer resultatet direkte i et element. Vil man have lidt mere kontrol, kan man benytte metoden ajax().

AJAX kræver jo normalt en eller anden dynamisk funktion på serveren, som kan kaldes af jQuery fra browseren, og i ASP.NET kan man eksempelvis vælge at bruge en service (ASMX/WCF), en page method (se side 202) eller en HTTP-handler.

I følgende eksempel bruges load() til at kalde en HTTP-handler (servertid.ashx), der returnerer serverens tid i ren tekst:

```
<html><head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {
   $.ajaxSetup({ cache: false });
   $("#aHentTid").click(function () {
    $("#spanResultat").load("ServerTid.ashx");
   });
  });
 </script>
</head>
<body>
 <a id="aHentTid" href="#">Hent server tid</a>:
 <span id="spanResultat"></span>
</body></html>
```

Når der klikkes på linket, placeres serverens tid i span-opmærkningen.

Figur 221. Brug af jQuery til AJAX-kald.

> En handler er en feature i ASP.NET, som blandt andet gør det muligt at slippe uden om page frameworket (ASPX-sider) og eksempelvis returnere andet end blot HTML. Den benyttes eksempelvis til at danne dynamiske billeder, dynamiske PDF-filer, CVS- (Excel) filer, eller som i førnævnte eksempel ren tekst. Udviklingen af handlers ligger uden for denne bogs rammer, men hvis man søger efter ”http handler asp.net”, vil man finde rigtig mange ressourcer.

jQueryUI

jQuery har, udover selve kernen som er netop beskrevet, yderligere et bibliotek kaldet jQueryUI. Det indeholder komponenter, som ikke er krævet for at få jQuery til at fungere, men derimod nye animationer og effekter, kontrollignende komponenter (i jQuery-terminologi kaldet widgets), samt en effektiv måde at ”style” disse widgets ved hjælp af et standardiseret bibliotek af CSS-filer.

Hvis man ønsker at bruge jQueryUI, skal der skabes en reference til filen:

```
jquery-ui-x.x.x.min.js
```

hvor x.x.x er selve versionen (i skrivende stund 1.8.6).

Filen kan findes på jquery.com, hvor det i øvrigt også er muligt at skabe en version af filen, som kun består af de komponenter, der skal bruges (i så fald vil filen hedde jquery-ui-x.x.x.custom.min.js)

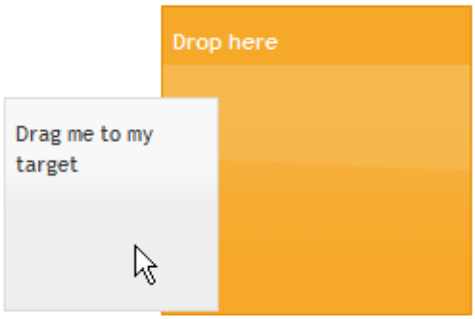
I biblioteket findes nye animationer til de fra kernen kendte metoder som show(), hide(), toggle(), addClass(), removeClass(), animate() med videre. Som eksempel på de forskellige animationer kan nævnes Bounce, Slide, Fold, Explode med videre (navnene er ret beskrivende for animationen). En “Bounce”-animation kan eksempelvis bruges som:

```
<html><head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript">
 </script>
 <script src="Scripts/jquery-ui-1.8.6.custom.min.js" type="text/javascript">
 </script>
 <script type="text/javascript">
  $(document).ready(function () {
   $("#btnKlik").click(function () {
    $(".demo").show("bounce");
   });
  });
 </script>
  <style type="text/css">
  .demo
```

```
  {
   display: none;
   padding: 10px;
   background-color: Yellow;
   border: 1px solid black;
   width: 200px;
  }
 </style>
</head>
<body>
 <input id="btnKlik" type="button" value="Klik på mig" /><br />
 <div class="demo">
  Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Curabitur
  consequat metus vitae est. Duis quis metus. Lorem ipsum dolor sit amet,
  consectetuer adipiscing elit. Quisque scelerisque.
</div></body></html>
```

Bemærk at der skabes en reference til jQueryUI, hvorefter show()-metoden kan kaldes med Bounce-animationen. Det giver ikke mening at vise et billede, fordi det er animationen, som er interessant, men prøv selv den viste kode eller se eksempler på jquery.com.

I jQueryUI findes ligeledes forskellige komponenter, som kan skabe en meget avanceret brugerflade. Der skal eksempelvis meget lidt kode til at skabe muligheden for at implementere drag'n drop.



Figur 222. Drag'n drop med jQueryUI (fra jquery.com).

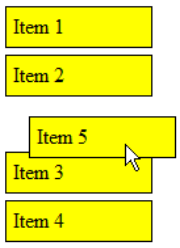
Der er ligeledes komponenter til at sortere lister, vise forskellige typer af dialogbokse, trække elementer rundt på brugerfladen og meget mere.

Her er et eksempel, der benytter metoden sortable() til at gøre det muligt at sortere en liste ved at trække rundt med elementerne på listen.

```
<html>
<head>
 <script src="Scripts/jquery-1.4.3.min.js" type="text/javascript">
 </script>
 <script src="Scripts/jquery-ui-1.8.6.custom.min.js" type="text/javascript">
 </script>
 <script type="text/javascript">
  $(document).ready(function () {
   $("#minListe").sortable();
  });
 </script>
 <style type="text/css">
  li
  {
   width: 100px;
   background-color: Yellow;
   border: 1px solid black;
   padding: 5px;
   margin: 5px;
   list-style-type: none;
  }
  ol
  {
   margin: 2px;
   padding: 2px;
  }
 </style>
</head>
<body>
 <ol id="minListe">
  <li>Item 1</li>
```

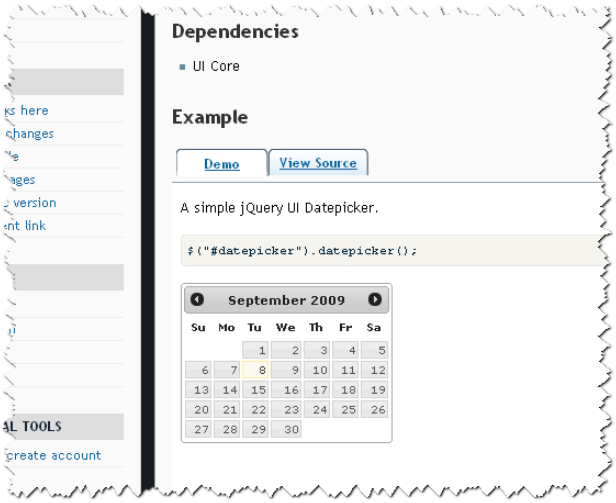
```
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
  <li>Item 5</li>
 </ol>
</body>
</html>
```

Bemærk hvor lidt kode (en linje) der kræves for at benytte en meget avanceret funktionalitet. Her er resultatet af siden (men igen – prøv det selv for at se funktionaliteten):



Figur 223. Sortering med jQueryUI.

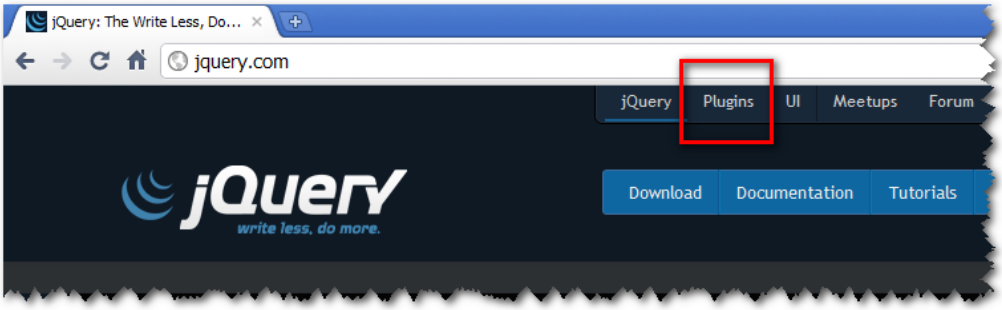
Slutteligt findes der otte såkaldte widgets, som kan bruges til eksempelvis at skabe forskellige typer af menuer, knapper, formularer med videre.



Figur 224. Eksempel på en jQuery-widget.

Plugins

Som det fremgår, er jQuery-bibliotekerne meget effektive til at skabe browseruafhængig JavaScript til at kontrollere brugerfladen i en browser, og man kan komme meget langt med selv få linjers script. I komplekse scenarier skal man dog – ligesom i alle andre teknologier – vide en del om platformen og hvordan man bedst benytter JavaScript, og derfor åbner jQuery mulighed for at inkludere kode fra andre. Denne feature kaldes plugins, og på jquery.com findes et helt område, hvor plugins fra personer og firmaer kan hentes.



Figur 225. Plugins kan findes på jquery.com.

I skrivende stund findes der et godt stykke over 1.000 plugins inden for kategorier som:

- AJAX
- Animationer
- Media
- Menu
- Navigation
- Tabeller
- Widgets
- … og mange flere

Set i lyset af at de fleste plugins er gratis og kan være skabt af nogen som ikke nødvendigvis kender jQuery til bunds, er der naturligt nok en svingende kvalitet, men der er rigtig mange plugins, som er meget brugbare. Man kan bruge en hel formiddag på at se på (og blive imponeret af) disse plugins, men her er nogle stykker, som man kan begynde med at se på:

Plugin	URL
DataTables: En plugin som kan konvertere en standard HTML-tabel til en tabel, som kan sorteres og filtreres (og meget mere) på klienten.	http://kortlink.dk/8bt6
jQuery Cycle Plugin: En plugin som viser billeder på en meget overbevisende måde.	http://kortlink.dk/8bt8
Easy Accordion: Flot menu-kontrol.	http://kortlink.dk/8bta

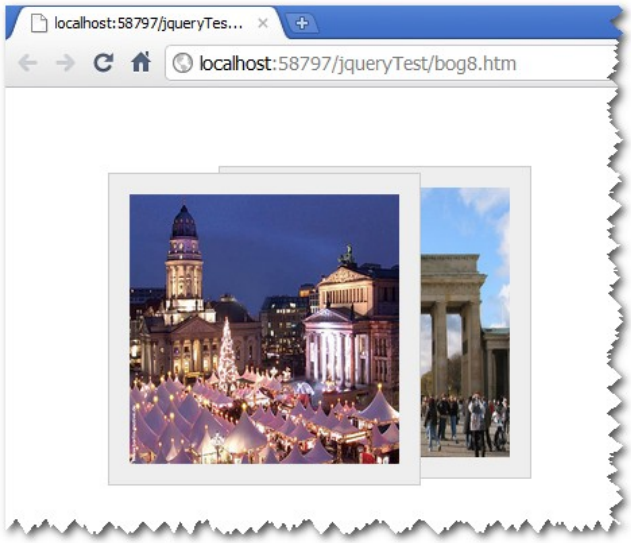
Tabel 33. Et par populære jQuery-plugins.

Ligesom ved brug af standard jQuery skal der sjældent særlig meget kode til at skabe en kompliceret brugerflade. Her er eksempelvis jQuery Cycle i funktion:

```
<html>
<head>
 <script src="Scripts/jquery-1.4.3.min.js"
  type="text/javascript"></script>
 <script src="Scripts/jquery.cycle.all.min.js"
  type="text/javascript"></script>
 <script type="text/javascript">
  $(document).ready(function () {
   $('#mineBilleder').cycle('shuffle');
  });
 </script>
 <style type="text/css">
  .billeder
  {
   top: 50;
   left: 150;
  }
  .billeder img
  {
   padding: 15px;
   border: 1px solid #ccc;
   background-color: #eee;
   width: 200px;
```

```
    height: 200px;
    top: 0;
    left: 0;
   }
  </style>
</head>
<body>
  <div id="mineBilleder" class="billeder">
   <img src="images/berlin.jpg" width="200" height="200" />
   <img src="images/berlin02.jpg" width="200" height="200" />
   <img src="images/brandenburgertor.jpg" width="200" height="200" />
  </div>
</body>
</html>
```

Plugin'en inkluderes på siden, hvilket giver mulighed for at kalde cycle()-metoden. Den skaber et smukt slideshow af billederne, som er angivet i div-opmærkningen (her bruges "shuffel"-animationen, men der er mange andre). Resten er bare CSS for at få en ramme om billederne med videre.



Figur 226. Cycle-plugin i funktion.

Man kan naturligvis også udvikle sine egne plugins, men det ligger uden for denne bogs rammer.