Cells

Il file 'cells.html' (trovate tutto in Cells.zip) realizza un gioco ispirato a ciò che è stato detto all'ultima riunione su agar.io. Il concetto è cellula grande mangia cellula piccola. Il giocatore controlla il movimento della cellula blu, il computer (ovvero il caso) controlla le altre cellule, rosse o cambiando il commento di colore casuale.
La parte html ha un titolo, una riga per punteggio e record e l'elemento canvas che consente di disegnare il campo da gioco e le cellule in movimento.
Il codice inizia con la definizione delle variabili, una serie di funzioni, la gestione dei tasti freccia e la chiamata alla funzione 'init' che crea la cellula del giocatore, le altre cellule ed il loop del gioco.
La funzione 'scena' controlla lo svolgimento del gioco, muove la cellula del giocatore in base ai tasti premuti e le altre cellule casualmente. Controlla se qualche cellula mangia altre cellule o esce dallo schermo. Quando una cellula mangia cresce, quando viene mangiata o esce dallo schermo esce dal gioco. Il giocatore vince quando rimane l'unica cellula o cresce troppo. Perde quando viene mangiato o esce dallo schermo. Ad ogni ciclo del gioco e ogni volta che una cellula viene mangiata, può nascere una nuova cellula.

CoderDojo Firenze

Filetto a 5

Il gioco:

Il filetto si gioca su una griglia ampia a piacere, e l'obiettivo è mettere in file 5 segni uguali.

Il programma:

Fase 1: disegno della griglia.

Il risultato della fase 1 è contenuto nel file filetto_1.html. La dimensione della griglia è definita dall'utente tramite 2 textbox che permettono di inserire il valore voluto da altezza e larghezza. Un bottone lancia la funzione JS che disegna la griglia con le dimensioni indicate.
La griglia è costituita da una tabella le cui celle sono gli elementi della griglia.
La tabella è caratterizzata dalle proprietà:
border="1" cellpadding="0" cellspacing="0"
che accostano le celle e le bordano con il bordo più sottile possibile.
Le celle vengono create dinamicamente nella funzione drawTable agganciandole alla classe CSS “griglia” che ne definisce dimensione in pixel, background, allineamento del testo, bordo e hover (quando il mouse passa sopra una cella si colora di giallo). Al termine delle funzione drawTable viene chiamata la funzione MostraNascondi per nascondere i textbox e il bottone.
La tabella è contenuta in un con id=”tutto”
Si fa uso delle variabili globali:
campo: array 2d destinata a contenere i valori delle celle
finito: booleano che gestisce il termine del gioco.

Classe CSS

```
<style>
.griglia{
	width: 25px;
	height: 25px;
	text-align: center;
	background-color: white;
}
.griglia:hover{
	background-color: yellow;
}
</style>
```

Funzione che disegna la griglia n colonne x m righe

```
function drawTable(n,m){
	x='<table id="tabella" border="1" cellpadding="0" cellspacing="0">';
```

```
	for (i=0; i<n; i++){
		x=x + '<tr>';
		campo[i]=[]; //il singolo elemento di campo è a sua volta una array
		for (j=0; j<m; j++){
			id= i +';'+j; //definisco un id univoco in funzione delle coordinate
			x=x + '<td class="griglia" id="' +id + '" >'+'</td>';
			campo[i][j]=-1;
		}
		x=x + '</tr>';
	}
	document.getElementById('tutto').innerHTML=x+'</table>';
	MostraNascondi('N');
	finito=false;
}
function MostraNascondi(x){
		if (x=='N') document.getElementById('grigliack').style.display='none';
		else document.getElementById('grigliack').style.display='block';
}
```

Elementi HTML

```
<body>
<div id="tutto">
</div>
</br><p id='grigliack'>
Larghezza griglia: <input type="text" id="larghezza" size="4" value="20">
Altezza griglia: <input type="text" id="altezza" size="4" value="20">
<input type="button" id="VIA" value="Comincia" onclick="drawTable
(document.getElementById('altezza').value,document.getElementById('larghezza').value);">
</p>
</body>
```

Fase 2: Scriviamo O e X

Si aggiunge una variabile globale che tiene conto del turno, e nella funzione drawTable viene aggiunta le gestione dell'evento onclick sulle celle della griglia. In pratica si aggancia il click su una cella della tabella alla funzione gestisciClick che scrive “X” o “O” nella cella in funzione del turno. Il risultato è nel file filetto_2.html.

Funzione drawTable (in giallo la modifica):

```
function drawTable(n,m){
	x='<table id="tabella" border="thin" cellpadding="0" cellspacing="0">';
	for (i=0; i<n; i++){
		x=x + '<tr>';
		campo[i]=[]; //il singolo elemento di campo è a sua volta una array
		for (j=0; j<m; j++){
			id= i +';'+j;
			x=x + '<td class="griglia"
onclick="gestisciClick('+i+','+j+','+n+','+m+')" id="' +id + '" >'+'</td>';
			campo[i][j]=-1;
		}
		x=x + '</tr>';
	}
	document.getElementById('tutto').innerHTML=x+'</table>';
	MostraNascondi('N');
	finito=false;
}
```

In pratica il click su ogni cella della tabella viene agganciato alla funzione `gestisciClick` che riceve come parametri gli indici i,j che rappresentano la posizione della cella all'interno della

griglia, e n,m che rappresentano le dimensioni della griglia.

La funzione gestisciClick si preoccupa di scrivere X o O in funzione del turno:

```
var turno='O';

function gestisciClick(i,j,n,m){
	if (finito) return;
	if (campo[i][j]==-1){
		campo[i][j]=turno;
		document.getElementById(i+';'+j).innerHTML=turno;
		turno=='O'?turno='X' : turno='O'
		document.getElementById('turno').value=turno;
	}
}
```

La funzione gestisciClick riceve al momento i parametri n,m senza usarli, li userà nella prossima fase. Si fa riferimento a un campo di tipo text con id='turno' in cui viene mostrato il valore della variabile turno per sapere a chi tocca.
Turno: <input type="text" id="turno" size="1" value="O" readonly>

Fase 3: chi vince?

A questo punto rimane solo da implementare la parte di codice che controlla quando ci sono 5 elementi uguali in fila. Il risultato è nel file definitivo filetto5.html. Questo viene fatto con la funzione checkWin che viene richiamata dalla funzione gestisciClick.
La funzione gestisciClick diventa:

```
function gestisciClick(i,j,n,m){
	if (finito) return;
	if (campo[i][j]==-1){
		campo[i][j]=turno;
		document.getElementById(i+';'+j).innerHTML=turno;
		if (checkWin(i,j,n,m,turno)==true){

document.getElementById('p_'+turno).value=parseInt(document.getElementById('p_'+turno).va
lue)+1;
			finito=true;
			MostraNascondi('M');
		}
		turno=='O'?turno='X' : turno='O'
		document.getElementById('turno').value=turno;
	}
}
```

In giallo sono riportate le modifiche. Viene fatto uso di due campi text per visualizzare il punteggio, i campo sono individuati con gli id 'p_O' e 'p_X'.

```
Punteggio O: <input type="text" id="p_O" size="4" value="0" readonly>
Punteggio X: <input type="text" id="p_X" size="4" value="0" readonly>
```

La funzione checkWin è composta di 4 parti, una per il controllo delle file orizzontali, una per le verticali, una per una diagonale e l'ultima per la diagonale opposta. Il controllo parte dalla posizione del click e guarda un intorno di 4 celle in orizzontale, verticale e sulle due diagonali. La funzione fa uso delle funzioni di utilità massimo e minimo, per verificare di non controllare celle inesistenti (fuori della tabella). Viene usata una array a che serve a memorizzare le celle che compongono la riga vincente che verranno colorate di verde.

```
function minimo(a,b){
	if (a<b) return a;
	return b;
}
function massimo(a,b){
	if (a>b) return a;
	return b;
}

function checkWin(pos_i,pos_j,n,m,turno){
	var a=[];
	if (finito) return false;
	start_i=massimo(pos_i-4,0);
	start_j=massimo(pos_j-4,0);
	stop_i=minimo(pos_i+5,n);
	stop_j=minimo(pos_j+5,m);
//Verticale
	somma=0;
	for (i=start_i; i<stop_i; i++){
		if (campo[i][pos_j]==turno){
			somma=somma+1;
			a.push(i+';'+pos_j);
			if (somma==5){
				for	(index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
				return true;
			}
		}
		else somma=0;
	}
//Orizzontale
	somma=0;a=[];
	for (j=start_j; j<stop_j; j++){
		if (campo[pos_i][j]==turno){
			somma=somma+1;
			a.push(pos_i+';'+j);
			if (somma==5){
				for	(index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
				return true;
			}
		}
		else somma=0;
	}
//Diagonale -45 deg
	somma=0;a=[];
	start=-minimo(pos_i-start_i, pos_j-start_j);
	stop=minimo(stop_i-pos_i,stop_j-pos_j);
	for (i=start; i<stop; i++){
		if (campo[pos_i+i][pos_j+i]==turno){
			somma=somma+1;
			a.push((pos_i+i)+';'+(pos_j+i));
			if (somma==5){
				for	(index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
				return true;
			}
		}
		else somma=0;
	}
//Diagonale 45 deg
	somma=0;a=[];
	start=-minimo(stop_i-pos_i-1, pos_j-start_j);
	stop=minimo(pos_i-start_i, stop_j-pos_j-1);
	for (i=start; i<stop; i++){
		if (campo[pos_i-i][pos_j+i]==turno){
```

```
                        somma=somma+1;
                        a.push((pos_i-i)+';'+(pos_j+i));
                        if (somma==5){
                                for     (index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
                                return true;
                        }
                }
                else somma=0;
        }
        return false;
}
```

Fase 4: e ora?

Si aprono diversi scenari di miglioramento:
- estetica, il gioco è essenziale, può essere abbellito sia permettendo di scegliere simboli diversi di O e X, sia migliorando le caselle di testo e il bottone.
- funzionale, facendo sì che la griglia cresca in base alle necessità, ovvero si potrebbe partire con una griglia 15x15 e ingrandirla ogni volta che viene fatto un click su una cella che dista meno di 5 elementi da uno dei bordi
- app, ovvero utilizzando un framework come phongap/cordova o analogo trasformare il gioco in una app
- multidevice, ovvero permettendo il gioco tra due persone ciascuna dal proprio device, realizzando un server di backend per gestire il gioco.
- multidevice peer-to-peer permettendo il gioco direttamente tra device senza avere un server

Appendice: Versione completa

File filetto5.html

```
<html>
<head>
<style>
.griglia{
        width: 25px;
        height: 25px;
        text-align: center;
        background-color: white;
}
.griglia:hover{
        background-color: yellow;
}

</style>
</head>
<script>
var campo=[];
var turno='O';
var finito=false;

function gestisciClick(i,j,n,m){
        if (finito) return;
        if (campo[i][j]==-1){
                campo[i][j]=turno;
                document.getElementById(i+';'+j).innerHTML=turno;
                if (checkWin(i,j,n,m,turno)==true){
//                      alert ('Ha vinto ' +turno );

document.getElementById('p_'+turno).value=parseInt(document.getElementById('p_'+turno).va
```

```
lue)+1;
                        finito=true;
                        MostraNascondi('M');
                }
                turno=='O'?turno='X' : turno='O'
                document.getElementById('turno').value=turno;
        }
}
function minimo(a,b){
        if (a<b) return a;
        return b;
}
function massimo(a,b){
        if (a>b) return a;
        return b;
}

function checkWin(pos_i,pos_j,n,m,turno){
        var a=[];
        if (finito) return false;
        start_i=massimo(pos_i-4,0);
        start_j=massimo(pos_j-4,0);
        stop_i=minimo(pos_i+5,n);
        stop_j=minimo(pos_j+5,m);
//Verticale
        somma=0;
        for (i=start_i; i<stop_i; i++){
                if (campo[i][pos_j]==turno){
                        somma=somma+1;
                        a.push(i+';'+pos_j);
                        if (somma==5){
                                for     (index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
                                return true;
                        }
                }
                else somma=0;
        }
//Orizzontale
        somma=0;a=[];
        for (j=start_j; j<stop_j; j++){
                if (campo[pos_i][j]==turno){
                        somma=somma+1;
                        a.push(pos_i+';'+j);
                        if (somma==5){
                                for     (index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
                                return true;
                        }
                }
                else somma=0;
        }
//Diagonale -45 deg
        somma=0;a=[];
        start=-minimo(pos_i-start_i, pos_j-start_j);
        stop=minimo(stop_i-pos_i,stop_j-pos_j);
        for (i=start; i<stop; i++){
                if (campo[pos_i+i][pos_j+i]==turno){
                        somma=somma+1;
                        a.push((pos_i+i)+';'+(pos_j+i));
                        if (somma==5){
                                for     (index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
                                return true;
                        }
                }
```

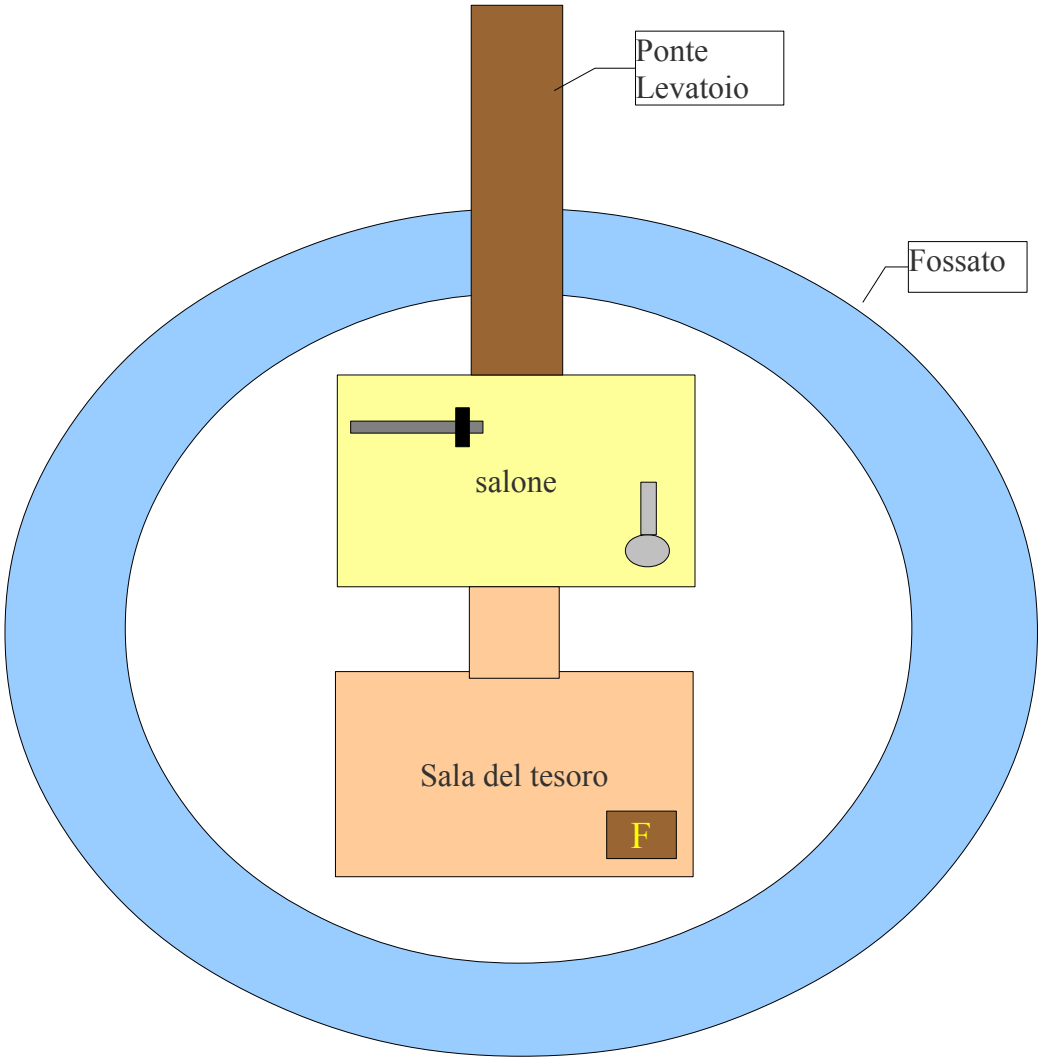
```
                else somma=0;
        }
//Diagonale 45 deg
        somma=0;a=[];
        start=-minimo(stop_i-pos_i-1, pos_j-start_j);
        stop=minimo(pos_i-start_i, stop_j-pos_j-1);
        for (i=start; i<stop; i++){
                if (campo[pos_i-i][pos_j+i]==turno){
                        somma=somma+1;
                        a.push((pos_i-i)+';'+(pos_j+i));
                        if (somma==5){
                                for     (index = 0; index < a.length; index++)

document.getElementById(a[index]).style.background= 'lightgreen';
                                return true;
                        }
                }
                else somma=0;
        }
        return false;
}
function MostraNascondi(x){
                if (x=='N') {
                        document.getElementById('grigliack').style.display='none';
                }
                else{
                        document.getElementById('grigliack').style.display='block';
                }
}

function drawTable(n,m){
        x='<table id="tabella" border="thin" cellpadding="0" cellspacing="0">';
        for (i=0; i<n; i++){
                x=x + '<tr>';
                campo[i]=[]; //il singolo elemento di campo Ã¨ a sua volta una array
                for (j=0; j<m; j++){
                        id= i +';'+j;
                        x=x + '<td class="griglia"
onclick="gestisciClick('+i+','+j+','+n+','+m+')" id="' +id + '" >'+'</td>';
                        campo[i][j]=-1;
                }
                x=x + '</tr>';
        }
        document.getElementById('tutto').innerHTML=x+'</table>';
        MostraNascondi('N');
        finito=false;
}
</script>
<body onload="MostraNascondi('M');">
<div id="tutto">
</div>
</br><p id='grigliack'>
Larghezza griglia: <input type="text" id="larghezza" size="4" value="20">
Altezza griglia: <input type="text" id="altezza" size="4" value="20">
<input type="button" id="VIA" value="Comincia" onclick="drawTable
(document.getElementById('altezza').value,document.getElementById('larghezza').value);">
</p>
Turno: <input type="text" id="turno" size="1" value="O" readonly>
Punteggio O: <input type="text" id="p_O" size="4" value="0" readonly>
Punteggio X: <input type="text" id="p_X" size="4" value="0" readonly>
<!-- per debug <p id="asd"></p>-->
</body>
</html>
```

```
<html>
<head>
    <!-- Gioco: palla si muove con frecce e mangia palla che si muove casualmente -->
    <script>
        var alto=128;
        var sinistra=128;
        var alto2=128;
        var sinistra2=128;
        var direz=Math.random()*6.28;
        var punteggio=0;
        var endgame=0;
        var alterna=0;
        window.addEventListener("keypress", function(event) { press(event); }, false);
        window.setInterval(function(){
            if (endgame==0){
                direz+=Math.random() * 0.7-0.35;
                sinistra2+=Math.cos(direz) *4;
                alto2-=Math.sin(direz)*4;
                if (sinistra2<0 || sinistra2>224 || alto2< 0 || alto2 > 224) direz+=3.14/2;
                sinistra2 = (sinistra2<0 ? 0:sinistra2);
                sinistra2 = (sinistra2>224 ? 224:sinistra2);
                alto2 = (alto2<0 ? 0:alto2);
                alto2 = (alto2>224 ? 224:alto2);
                document.getElementById("palla2").style.left=sinistra2.toString()+'px';
                document.getElementById("palla2").style.top=alto2.toString()+'px';
                if (alterna==0){
                    document.getElementById("palla2").style.backgroundImage='url("palla5.png")';
                    alterna=1;
                }else{
                    document.getElementById("palla2").style.backgroundImage='url("palla6.png")';
                    alterna=0;
                }
            }
        },80);
        window.setInterval(function(){
            if (endgame==0){
                endgame=1;
                alert('Game Over');
            }
        },30000);

        function press(e){
            if (endgame==0){
                var eventReference = (typeof e !== "undefined")? e : event;
                var keyCode = eventReference.keyCode;
                step=4;
                switch (keyCode){
                    case 37: //sinistra
                            sinistra-=step;
                            sinistra = (sinistra<0 ? 0:sinistra);
                        break;
                    case 38: //alto
                            alto-=step;
                            alto = (alto<0 ? 0:alto);
                        break;
                    case 39: //destra
                            sinistra+=step;
                            sinistra = (sinistra>224 ? 224:sinistra);
                        break;
                    case 40: //basso
                            alto+=step;
                            alto = (alto>224 ? 224:alto);
                        break;
                }
                if (((alto-alto2)*(alto-alto2)+(sinistra-sinistra2)*(sinistra-sinistra2))<16){
                    punteggio++;
                    document.getElementById("punteggio").innerHTML=punteggio;
                    alto=0; sinistra=0;
                }
                document.getElementById("palla").style.left=sinistra.toString()+'px';
                document.getElementById("palla").style.top=alto.toString()+'px';
            }
        }
    </script>
</head>
<body>
<div  id="campodigioco" style="height:256;width:256;border:solid 1px black">
    <div id="palla" style="background-image:url
('palla.png');position:absolute;top:128px;left:128px;width:32px;height:32px;"></div>
    <div id="palla2" style="background-image:url
('palla.png');position:absolute;top:128px;left:128px;width:32px;height:32px;"></div>
</div>
Punteggio:<label id="punteggio">0</label>
</body></html>
```


Ponte
Levatoio
Fossato
salone
Sala del tesoro
F

Using SVG with HTML5 tutorial

From EduTech Wiki

Page created by Daniel K. Schneider, 1 April 2012
Last modified by Daniel K. Schneider, 19 February 2014

Contents

1 Introduction

According to Wikipedia (http://en.wikipedia.org/wiki/Scalable_Vector_Graphics) (retrieved April 1 2012), “Scalable Vector Graphics (SVG) is a family of specifications of an XML-based file format for two-dimensional vector graphics, both static and dynamic (i.e. interactive or animated). The SVG specification is an open standard that has been under development by the World Wide Web Consortium (W3C) since 1999. SVG images and their behaviors are defined in XML text files. This means that they can be searched, indexed, scripted and, if required, compressed. As XML files, SVG images can be created and edited with any text editor, but it is often more convenient to create them with drawing programs such as Inkscape. All major modern web browsers have at least some degree of support and render SVG markup directly, including Mozilla Firefox, Internet Explorer 9, Google Chrome, Opera and Safari. Earlier versions of

Microsoft Internet Explorer (IE) do not support SVG natively.”.

SVG is part of the HTML 5 draft specification, i.e. SVG tags are part of the language and can be inline. We have seen it working since Chrome 9 and 10, Firefox 4, Opera 11 and Internet Explorer 9 (feb 2011). Support of SVG is still not complete, but rapidly improving. All major browser brands (except IE9) now offer very good support - Daniel K. Schneider 19:47, 1 April 2012 (CEST)

If you want to know whether your browser can handle SVG and HTML5, look at this See the When can I use Inline SVG in HTML5? (http://caniuse.com/svg-html5) compatibility table at caniuse.com.

Further reading:

- Using Inkscape for web animation. This article includes some additional tips on using Inskcape and repurposing complex drawings found on openclipart.org

See also:

- SVG (very short overview article)
- SVG links (nothing but mostly good links)
- Static SVG tutorial
- SVG/SMIL animation tutorial
- Interactive SVG-SMIL animation tutorial
- XSLT to generate SVG tutorial

2 Learning and reusing SVG

2.1 Learning SVG

The SVG links article includes a number of good links. Otherwise, start from the Static SVG tutorial.

If you want to dive in, use:

- SVG-edit (http://svg-edit.googlecode.com/svn/trunk/editor/svg-editor.html) (online editor at googlecode)

Respect the syntax of the SVG fragment:

Minimal example (not recommended):

```
<svg xmlns="http://www.w3.org/2000/svg">
    <circle id="greencircle" cx="30" cy="30" r="30" fill="green" />
</svg>
```

Unfinished example that defines a SVG canevas size. By default width and height are pixels. You also could use cm's, mm's etc.

```
<svg height="200" with="600" xmlns="http://www.w3.org/2000/svg">
```

```
      <!-- SVG tags go in here, between a few lines and several hundreds ..
</svg>
```

Unfinished example enabling links (needed for animations/interactive pages!)

```
<svg height="6cm" width="10cm">
   xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/x
   <!-- SVG tags go in here, between a few lines and several hundreds ....
</svg>
```

If you want to draw a border, use standard HTML CSS properties, e.g. like this:

```
<svg style="border-style:solid;border-width:1px;" id="circle" height="60" w
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
     <!-- SVG tags go in here, between a few lines and several hundred ....
  </svg>
</svg>
```

2.2 Cleaning up SVG

... is difficult. You may try to import a file into an SVG editor and save again. So far, no editor can handle HTML5 with embedded SVG.

If your SVG file doesn't display because of syntax errors, try the W3C validation service. It will tell you what is wrong in which lines:

- http://validator.w3.org/
- You also may submit an HTML5 file with embedded SVG. Make sure to include the right doctype definition on top, ie. <!doctype html>

Coping with Inkscape drawings

Using drawings made with Inkscape very often require extra work since fairly weird coordinates may have been used. Chapter 23. SVG and the Web (http://tavmjong.free.fr/INKSCAPE/MANUAL/html/Web.html) chapter of the official manual provides some tips. In particular, read Inkscape for the Web (http://tavmjong.free.fr/INKSCAPE/MANUAL/html/Web-Inkscape.html). The most important tip is to **Save as Optimize SVG**. This operation may set a viewBox if none is present and will set the width and height to 100%. This allows to display the full picture even if you make it small.

Suggested workflow (adapt to your needs) for making a typical Inkscape document web ready:

1. **Fit the document size to the picture size:** Menu File->Document Properties. In the Custom size panel, open **Resize page to content**, then click **Resize page to drawing or selection**.
2. Remove unused DEF's: File->Vacuum Defs.
3. **Optimize SVG'*: File -> Save as... -> Optimized SVG*.** *Tick "Enable viewboxing* in the save dialog. This operation will do several things, in particular, insert correct viewBox, width and height attributes.
4. **Remove non-standard SVG/Inkscape XML**: File -> Save as... -> plain SVG (Since this operation will remove layers and other Inkscape specific

information, keep a copy of the old file if necessary). As of Jan 2014, this can trigger a display error as shown in Using Inkscape for web animation. Reopen the saved file and the blackness may be gone ....

Again, consider keeping a copy of the original drawing. This procedure mail fail, in particular when drawings were made with non standard SVG ! You also can try to permute order, i.e. save as plain SVG first before you optimize.

Openclipart.org drawings

Most of these drawings were probably made with Inkscape and "posted as is" and you may have to clean up a bit as explained above. However, most drawings will work just fine. In addition, some SVG drawings at http://openclipart.org import just fine, e.g. the coconut tree (http://openclipart.org/detail/150253/palm-tree-by-newt) do not include proprietary extensions. Notice that these are not illegal, the web browser just can and will ignore these.

```
<?xml version="1.0"?>
  <svg xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
       xmlns="http://www.w3.org/2000/svg"
       xmlns:cc="http://creativecommons.org/ns#"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:svg="http://www.w3.org/2000/svg" id="svg2"
       viewBox="0 0 744.09 1052.4" version="1.1">
</svg>
```

This file doesn't include any proprietary Inkscape extensions. It just includes some extra vocabularies that is used to include various meta information (description, copyright, etc.). Below a file that shows both the original and a cleanup version. There isn't much difference, since we didn't remove extra space around the drawing, it appears a bit smaller.

Have a look:

- http://tecfa.unige.ch/guides/svg/ex/html5/inkscape-import/html5-import-with-img.html

Below is another example:

- http://tecfa.unige.ch/guides/svg/ex/html5/inkscape-import/html5-import-with-img2.html

3 How can I include SVG in HTML5

There are several ways of including SVG in HTML5. So-called inlining is most often preferred. However, inlining complex graphics makes hand-coding HTML a pain since the file can grow quite a lot.

In this chapter we just will show how to embed SVG images that are the right size. Basically, any sort of "inclusion" mechanism will work:

- Copy/paste SVG code within HTML code (inlining)

- Using the HTML *img* tag
- Using the HTML *object* tag
- Using the HTML *iframe* tag
- Using CSS (background images)
- Including SVG within SVG using the *image* tag.

Further below we shall explain how to adapt the size of embedded SVG images.

3.1 Inlining SVG in HTML 5

Do **not** forget the namespace declaration(s) in the top-level svg element. At least: `xmlns="http://www.w3.org/2000/svg"`

Live example:

- circle.html (http://tecfa.unige.ch/guides/svg/ex/html5/circle.html) (look at the source)

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
  </head>

  <body>
    <h1>HTML5 SVG Demo</h1>

A nice green circle:
    <svg id="circle" height="200" xmlns="http://www.w3.org/2000/svg">
      <circle id="greencircle" cx="30" cy="30" r="30" fill="green" />
    </svg>

    <hr>
    <address>Created by DKS. This is free code</address>
  </body>
</html>
```

Live example of an animated SVG:

- circle-crawling.html (http://tecfa.unige.ch/guides/svg/ex/html5/circle-crawling.html) (look at the source)

Notice:

- SVG also can import SVG through using its "images" tag. However, this is only supposed to work with static SVG. This methods is introduced in the next chapter

3.2 Embeding SVG in HTML 5 with the img tag

This only works with **static SVG**. Use the `object` element or the *SVG* image element if your SVG includes animations and/or interactive elements.

Live example:

- html5-with-img-src.html (http://tecfa.unige.ch/guides/svg/ex/html5/html5-with-img-src.html)

Below are the most important elements of this example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>HTML5 SVG Demo (embed with img src)</h1>

   <p> A nice green circle that was embeded using the img element:
   <img src="green-circle.svg" height="64" alt="Nice green circle"/>
</p>
```

3.3 Embeding SVG in HTML 5 with the object tag

Live example:

- html5-with-object.html (http://tecfa.unige.ch/guides/svg/ex/html5/html5-with-object.html)

Below are the most important elements of this example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>HTML5 SVG Demo (embed with object)</h1>

<p> A nice green circle that was embeded using the HTML "object" tag:
<object type="image/svg+xml" data="green-circle.svg"
        width="64" height="64" border="1"></object>
</p>
</body>
</html>
```

3.4 Embeding SVG in HTML 5 with the iframe tag

Live example:

- html5-with-iframe.html (http://tecfa.unige.ch/guides/svg/ex/html5/html5-with-iframe.html)

Below are the most important elements of this example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>HTML5 SVG Demo (embed with iframe)</h1>

<p> A nice green circle that was embeded using the HTML "iframe" tag:
<iframe src="green-circle.svg"
        width="64" height="64" style="border:1;"></iframe>
</p>
<p>
Tips:
<iframe src="green-circle.svg" style="float:left;margin-right:1cm;"
        width="64" height="40" style="border:1;"></iframe>
</p>
</body>
</html>
```

3.5 Using SVG as background image in HTML 5

Below are the most important elements of this example: Live example:

- html5-with-css-background.html (http://tecfa.unige.ch/guides/svg/ex/html5/html5-with-css-background.html)

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
    <style type="text/CSS">
      body {
      background-image: none, url('green-circle.svg');
      background-position: 50% 50%;
      background-repeat: no-repeat;
      }
    </style>
  </head>

  <body>
    <h1>HTML5 SVG Demo (embed with CSS background)</h1>

<p> A nice green circle that was embeded using CSS Background.
</p>
<p>Tips:</p>
<ul>
<li>SVG included as CSS can't be scripted and positioning is hard. Therefor
</ul>
</body>
</html>
```

3.6 Including SVG through the SVG image element

Make sure that the included SVG defines its size in terms of percentages or that you set a viewBox.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <h1>HTML5 SVG Demo - embed svg file with SVG image</h1>

<p> A huge red circle that was embeded using the svg "image" tag:</p>
<svg id="circle" height="60" width="60"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <image x="0" y="0" height="60" width="60"  xlink:href="huge-red-circle.sv
</svg>
</body>
```

- Life example: import-with-svg-image.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/import-with-svg-image.html)

4 Adapting the size and position of an SVG graphic

There exist various way of doing this. Our preferred method for static SVG images is to fix the original SVG graphic with a viewbox and size attributes and then import with the HTML **img** tag. The original SVG should look like this, i.e. include a viewBox that uses the original width and size of the drawing (to be adjusted to your drawing of course), and width and height set to 100%.

Fixing the original SVG code (see below how to do this with Inkscape)

```
<svg
   xmlns="http://www.w3.org/2000/svg"
   version="1.1"
   width="100%"
   height="100%"
   viewBox="0 0 684 648">
```

You then can simply set a height in your HTML code, e.g.

```
<img src="coconut.svg" height="100" alt="Nice coconut tree"/>
```

Life example:

- html5-import-with-img.html (http://tecfa.unige.ch/guides/svg/ex/html5/inkscape-import/html5-import-with-img.html)

- Our second preferred method is to import the SVG graphic with its original size and then wrap it inside a scale transform as explained in more detail further down. Another method that also works very well is to fix the original SVG drawing as explained in the section about Inkscape and then just use the simple HTML *image* element.

```
<svg id="sir" height="116" width="106"
xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlin
<g transform="scale(0.1)">
  <image x="0" y="0" height="1140" width="1040"  xlink:href="like_a_sir_ori
</g>
</svg>
```

4.1 Using the SVG transform attribute to change the size of a graphic

For starters, we suggest editing the SVG file and adding the following "transformation" code:

Right after the opening svg tag add this:

```
  <g transform="scale(0.1)">
```

Right before the closing svg tag add this:

```
  </g>
```

You can use this method within your SVG graphic in order to adapt the size of various parts. Just wrap them inside a *g transform="scale(...)"*.

Success rate: This method is guaranteed to work since it relies on basic SVG technology that is now over 10 years old. However, this is not the most elegant solution since it requires editing the SVG file. See the next solution.

Simple live example:

- circle-scale.html (http://tecfa.unige.ch/guides/svg/ex/html5/circle-scale.html) that you should compare with circle.html (http://tecfa.unige.ch/guides/svg/ex/html5/circle.html)

The simple SVG file we are going to import in the following examples is huge-red-circle.svg (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/huge-red-circle.svg). It looks like this:

```
<?xml version="1.0"?>
<svg height="600" width="600" xmlns="http://www.w3.org/2000/svg">
  <circle cx="50%" cy="50%" r="50%" fill="red" />
</svg>
```

4.2 Using the SVG image tag to import and adapt an SVG graphic

The next methods are fairly elegant, easy to understand and should work in all browsers. There are three variants. For now, I suggest to use variant B - DKS.

4.2.1 The simple solution

Let's recall how to use the SVG image tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <h1>HTML5 SVG Demo - embed svg file with SVG image</h1>

<p> A huge red circle that was embeded using the svg "image" tag:</p>
<svg id="circle" height="60" width="60"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <image x="0" y="0" height="60" width="60"  xlink:href="huge-red-circle.sv
</svg>
</body>
```

- Life example: import-with-svg-image.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/import-with-svg-image.html)

All you typically have to do is:

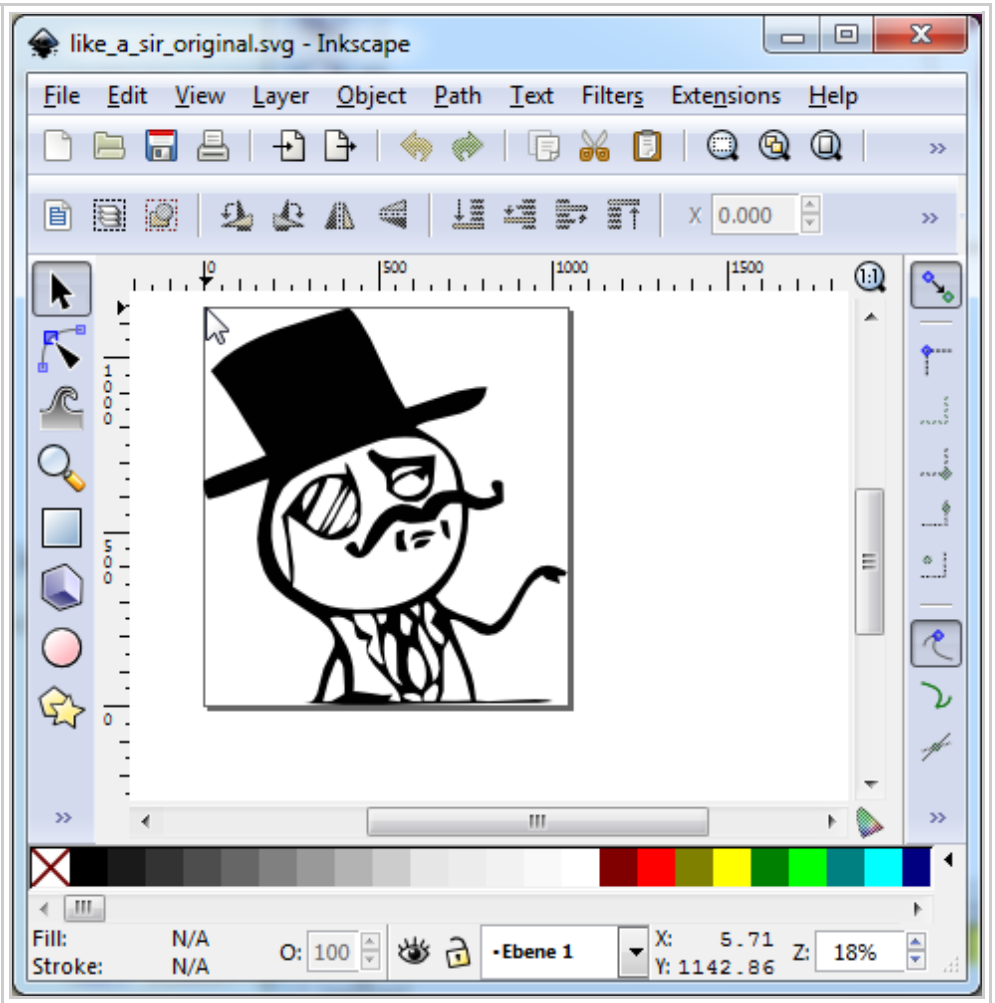
- Figure out the proportions of the original SVG (look at the svg root element of the SVG file)
- Define the height and width of the image with respect to your scaling needs.

In our example, the imported graphic's dimensions were 600 x 600 px. In order to get a circle 10 times smaller, we just dived x/10 and y/10.

- If you can't see the picture, import the picture in its original size and set the canvevas size to the same dimensions. E.g. if the original canevas is 1000x400:

```
<svg id="circle" height="1000.7" width="400"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <image x="0" y="0" height="1000.7" width="400"  xlink:href="huge-red-circ
</svg>
```

Now that you know that the picture and your code is ok, you may try to understand why you may not see a picture. Indeed, this simple method is **not** guaranteed to work with every SVG file. In particular it will not when the imported image is "badly" positioned. See the next solution, using a scale transform of the imported SVG, if don't want to learn how to fix that kind of problem in the imported SVG code. Let's illustrate this with an example. The like a Sir (http://openclipart.org/detail/168634/like-a-sir-by-tavin) picture is positioned in negative y space as you can see in the following screenshot and code excerpt:



SVG picture in negative space

```
<g inkscape:label="Ebene 1" inkscape:groupmode="layer" id="layer1" transfor
```

Solutions:

- Open the SVG file with Inkscape and fix it. This way you also could get rid of all the extension tags (save as "plain SVG"). However, this may not be easy. The graphic is likely to be in a wrong position, because the author created paths with SVG path data that do not fit the "usual" x=0/y=0 coordinate system. Something one never should do ....
- Use one of the methods below. That is: **import the picutre** with its original size, then scale it either with a scale transform (easier) or by adjusting the ViewBox.

Of course, you can move pictures The following shows how to reposition a "normal" image.

```
<svg id="circle" height="62" width="62"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <image x="1" y="1" height="60" width="60"  xlink:href="huge-red-circle.sv
</svg>
```

4.2.2 Import the graphic with its original size and scale the image

Below is another variant that seems is more robust with respect to drawings that use strange coordinates. We import the original huge circle with its original size, but we then scale transform it.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <h1>HTML5 SVG Demo - embed svg file with SVG image</h1>

<p> A huge red circle that was embeded using the svg "image" tag plus a sca
<svg id="circle" height="60" width="60"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <g transform="scale(0.1)">
    <image x="0" y="0" height="600" width="600" xlink:href="huge-red-circle
  </g>
</svg>
</body>
</html>
```

Pay attention to the following tips:

- Height and width of the image tag **must** correspond (roughly) to the size of the imported SVG. Look at its SVG root tag. If weight and height are missing in the SVG file add it yourself, if possible
- Height and width of the svg root tag should somehow fit the scaled image. E.g. if you import a 100x200 and make it twice as big, then svg root dimensions should be at least 200x400.

- Life example: import-with-svg-image2.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/import-with-svg-image2.html)

- Life example2: like_a_sir.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/like_a_sir.html)

4.2.3 Importing with original size and create a viewBox

This solution also offers a way to rescale. viewBox attributes can do more, but we won't cover that.

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>HTML5 SVG Demo - embed svg file with SVG image and viewBox</h1>

<p> A huge red circle that was embeded using the svg "image" tag plus a cha
```

```
<svg id="circle" height="60" width="60"
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <svg viewBox="0 0 600 600">
    <image x="0" y="0" height="600" width="600" xlink:href="huge-red-circle
  </svg>
</svg>
</body>
</html>
```

- Life example: import-with-svg-image3.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/import-with-svg-image3.html)

Playing with viewBox

Possibilities are endless, but viewBox is hard to understand ...

The following code shows how to stretch and reposition (move the viewport up)

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML5 SVG demo</title>
    <meta http-equiv="Content-Type" content="text/html;charset=utf-8" />
  </head>

  <body>
    <h1>HTML5 SVG Demo - embed svg file with SVG image and distort with vie

<p> A huge red circle that was embeded using the svg "image" tag plus a cha
<svg style="border-style:solid;border-width:1px;" id="circle" height="60" w
     xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999
  <svg viewBox="0 -300 600 1200" preserveAspectRatio="none" >
    <image x="0" y="0" height="600" width="600" xlink:href="huge-red-circle
  </svg>
</svg>
</body>
</html>
```

- Life example: import-with-svg-image4.html (http://tecfa.unige.ch/guides/svg/ex/html5/svg-import/import-with-svg-image4.html)

4.3 Creating text that floats around SVG images

You can use ordinary HTML CSS properties to style the svg element. The following SVG graphic will float to the right:

```
<svg style="float:right" width="81" height="127" xmlns="http://www.w3.org/2
 <!-- Created with SVG-edit - http://svg-edit.googlecode.com/ -->
 <g>
  <title>Simple tree</title>
  <rect fill="#7f3f00" stroke="#000000" stroke-width="0" x="32.26172" y="50
  <circle fill="#007f3f" stroke="#000000" stroke-width="0" cx="40.26172" cy
 </g>
</svg>
```

The following code would make an image float to the left and add a small margin:

```
<svg width="43" height="65" style="float:left;margin:5px;"
    xmlns="http://www.w3.org/2000/svg">
....
</svg>
```

Live example that includes some long SVG code (look at the source):

- text-with-pics.html (http://tecfa.unige.ch/guides/svg/ex/html5/text-with-pics/text-with-pics.html)
- To do: Create a version that uses the SVG image tag for importing.

5 Links

Specifications (very technical reading !)

- The image element (http://www.w3.org/TR/SVG/struct.html#ImageElement) (part of the Scalable Vector Graphics (SVG) 1.1 (http://www.w3.org/TR/SVG/Overview.html) (Second Edition), W3C Recommendation 16 August 2011
- HTML5 A vocabulary and associated APIs for HTML and XHTML (http://dev.w3.org/html5/spec/single-page.html), W3C Working Draft 29 March 2012. Have a look at the chapter on Embedded content (http://dev.w3.org/html5/spec/single-page.html#embedded-content-1)

ViewBox

- 7.7 The ‘viewBox’ attribute (http://www.w3.org/TR/SVG11/coords.html#ViewBoxAttribute) (SVG 1.1 Specification)
- SVG Viewport and View Box (http://tutorials.jenkov.com/svg/svg-viewport-view-box.html)

CoderDojo Firenze

Gioco a bivi: Avventura nel castello

Si trae ispirazione dal gioco in 5 righe di html, diffuso dal Dojo di Galway. Si tratta di costruire un percorso a scelte multiple tramite link a diverse pagine html. Inizialmente si dà priorità alla logica del gioco usando pochi tag html e nessun elemento di stile, per inserire successivamente ulteriori elementi html e di stile che permetteranno di abbellire le pagine.

Passo 1: Logica del gioco

È il momento di creare una cartella per contenere i vari file che saranno necessari, dopodiché si parte realizzando la pagina index.html con il seguente codice:

index.html:

```
<html>
<body>
<p>Sei davanti a un castello circondato da un fossato pieno di coccodrilli.<br/>
Puoi <i>entrare</i> nel castello o <i>saltare</i> nel fossato<br/>
<img src="castello.jpg" /></p>
        <a href ="entra.html">Entra</a> o <a href="salta.html"> Salta </a>
</body>
</html>
```

Questa pagina dà due scelte, di passare alla pagina entra.html, o alla pagina salta.html che sono le seguenti:

entra.html:

```
<html>
<body>
<p>Sei dentro al castello<br/>
Complimenti</p>
<img src="sala.jpg" /><br/>
        <a href ="index.html">Torna Fuori</a>
</body>
</html>
```

salta.html:

```
<html>
<body>
<p>Sei stato un ottimo pasto per il coccodrillo, che ti ringrazia con affetto</p>
<img src="coccodrillo.jpg" /><br/>
        <a href ="index.html">Ricomincia</a>
</body>
</html>
```

Le pagine fanno riferimento a immagini che sono nella stessa cartella dei file html, e devono ovviamente essere già state predisposte.
A questo punto i ragazzi possono estendere o complicare la logica aumentando il numero di scelte, e il numero di livelli.

Passo 2: abbellimento grafico

Si inseriscono sia nuovi tag html, che elementi di stile, in particolare:
<head>, <title>, <style>, <h1> come tag;
diversi elementi di stile per gestire colore dei font, colore di sfondo e caratteristiche grafiche dei link.

index.html diventa:

```
<html>
<head>
	<title>Avventura nel castello</title>
	<style>
		p {	color: white;
			font-size: 16px;
		}
		a {	text-decoration: none;
			color: blue;
		}
	</style>
</head>
<body style="background-color:black">
	<h1 style="color:lightblue">Avventura nel castello</h1>
	<p>Sei davanti a un castello circondato da un fossato pieno di coccodrilli.<br/>
Puoi <i>entrare</i> nel castello o <i>saltare</i> nel fossato<br/>
	<img src="castello.jpg" /></p>
	<p><a href ="entra.html">Entra</a> o <a href="salta.html">Salta</a></p>
</body>
</html>
```

Su questa falsariga si modificano entra.html e salta.html, facendo usare le funzioni di copia e incolla. Si mette in evidenza che è sufficiente copiare e incollare il tag <style> per replicare le caratteristiche grafiche della pagina index.html sulle altre

entra.html:

```
<html>
<head>
	<title>Avventura nel castello - Hai vinto!</title>
	<style>
		p { color: white;
			font-size: 16px;
		}
		a {	text-decoration: none;
			color: blue;
		}
	</style>
</head>
<body style="background-color:black">
	<h1 style="color:green">Avventura nel castello - Hai vinto!</h1>
	<p>Sei dentro al castello<br/>
Complimenti</p>
	<img src="sala.jpg" /><br/>
	<a href ="index.html">Torna Fuori</a>
</body>
</html>
```

salta.html:

```
<html>
<head>
	<title>Avventura nel castello - Hai perso!</title>
	<style>
		p { color: white;
			font-size: 16px;
		}
		a {	text-decoration: none;
			color: blue;
		}
	</style>
</head>
<body style="background-color:red">
	<h1 style="color:black">Avventura nel castello - Hai perso!</h1>
	<p>Sei stato un ottimo pasto per il coccodrillo, che ti ringrazia con affetto</p>
	<img src="coccodrillo.jpg" /><br/>
	<a href ="index.html">Ricomincia</a>
</body>
</html>
```

Si vede come le impostazioni generali di stile possano essere definite nella sezione <head>, inserendole una sola volta, mentre le impostazioni specifiche di un tag vadano inserite nel tag stesso, come nei tag <h1> e <body>.

CoderDojo Firenze

JSON: JavaScript Object Notification

Permette di rappresentare dei dati in un formato semplice e immediato adatto allo scambio dati tra applicazioni come quello necessario per la programmazione AJAX (Asincronous Javascript and XML).
I dati rappresentabili sono:
- boolean
- numeri
- stringhe (delimitate da “)
- array (separati da , e racchiusi da [])
- array associativi, ovvero array di coppie chiave/valore racchiusi da {}
- null

JSON è basato su due strutture:

Un insieme di coppie nome/valore. In diversi linguaggi, questo è realizzato come un oggetto, un record, uno struct, un dizionario, una tabella hash, un elenco di chiavi o un array associativo.
Un elenco ordinato di valori. Nella maggior parte dei linguaggi questo si realizza con un array, un vettore, un elenco o una sequenza.

Il linguaggio JavaScript prevede metodi che trasformano variabili in JSON e viceversa.
Un esempio di JSON è il seguente:

```
{
    "titolo": "Questo è un JSON",
    "valore": "esempio di valore",
    "elementi": [
        {"chiave1_el1": "valore1", "chiave2": "valore2"},
        {"chiave1_el2": "valore3", "chiave2_el2": "valore4","chiave3_el2": "valore5"}
    ]
}
```

In Javascript Per convertire una stringa JSON in oggetto è possibile usare il seguente codice:

```
var myObject = eval('(' + myJSONtext + ')');
```

per proteggersi da exploit possibili attraverso la funzione eval() meglio però usare

```
var myObject = JSON.parse(myJSONtext, reviver);
```

dove reviver è parametro opzionale e rappresenta un puntatore a funzione che viene chiamata per ogni coppia chiave/valore presente nella string JSON.

Viceversa per ottenere una stringa JSON da un oggetto Javascript è possibile usare:

```
var myJSONText = JSON.stringify(myObject, replacer);
```

dove replacer è un parametro opzionale che viene chiamato per ogni coppia chiva/valore e consente di effettuare una conversione di valori.