ANIMACIÓN

WEB

THE ILLUSION OF LIFE

EL LIBRO, *THE ILLUSION OF LIFE: DISNEY ANIMATION* ES CONOCIDO COMO LA BIBLIA DE ANIMACIÓN ENTRE ARTISTAS Y ANIMADORES DE TODO EL MUNDO. EN EL SE EXPLICAN LOS 12 PRINCIPIOS DE ANIMACIÓN.

EL DISEÑADOR CENTO LODIGIANI LOS HA SIMPLIFICADOS A UN CUBO PARA ENTENDERLOS MEJOR. VAMOS A VER LOS 4 MÁS APLICABLES EN WEB.

ESTIRAR Y ENCOGER

LA EXAGERACIÓN Y LA DEFORMACIÓN DE LOS CUERPOS. UN CUERPO DEBE CONSERVAR SU VOLUMEN, Y SEGÚN SE ESTIRE Y ENCOJA NOS DA SENSACIÓN DE VOLUMEN Y PESO. Y AL MISMO TIEMPO NOS PUEDE AYUDAR A DAR UN EJFECTO MÁS CÓMICO.

ANTICIPACIÓN

SE DEBEN ANTICIPAR LOS MOVIMIENTOS PARA GUIAR LA MIRADA DEL USUARIO Y ANUNCIAR LO QUE VA A PASAR.

SLOW IN Y SLOW OUT

LOS OBJETOS NECESITAN TIEMPO PARA ACELERAR Y REDUCIR LA VELOCIDAD. ES ESPECIALMENTE ÚTIL PARA DAR CREDIBILIDAD A LAS ANIMACIONES.

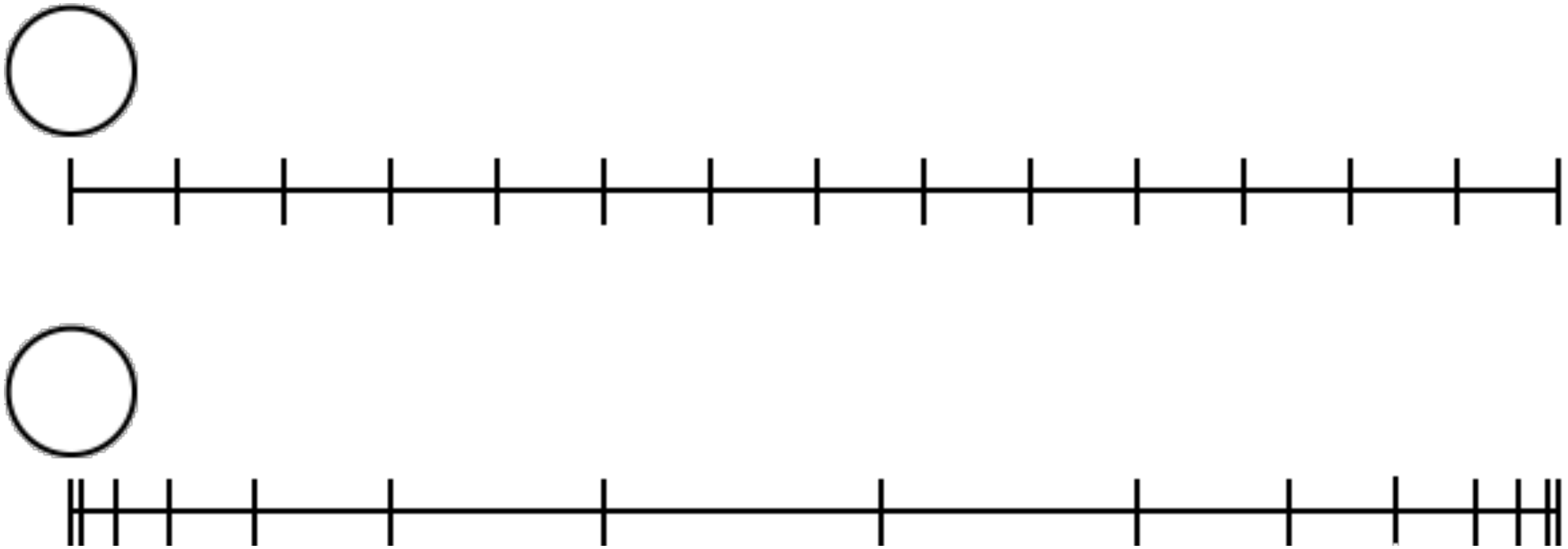
ACCIÓN SECUNDARIA

CONSISTE EN PEQUEÑOS MOVIMIENTOS QUE COMPLEMENTAN A LA ACCIÓN PRINCIPAL SIN ESTAR MÁS MARCADA QUE LA ACCIÓN PRINCIPAL. PERMITE DETERMINAR UNA JERARQUÍA VISUAL.

¿POR QUÉ ANIMACIÓN EN WEB?

- DA UN ASPECTO MEJOR Y MÁS PROFESIONAL.
- NOS AYUDA CON LA UI/UX, APOYANDO AL DISEÑO.
- INTENTAR QUE TODO SEA NATURAL Y FAMILIAR PARA EL USUARIO.
- NO FORZAR AL USUARIO A PENSAR QUE HA OCURRIDO.
- GUIAR AL USUARIO AL SIGUIENTE PASO.
- CENTRAR AL USUARIO A QUE COSAS PUEDE HACER.
- CON MENOS TEXTO TODO SE ENTIENDE MEJOR
- SPA SIN ANIMACIÓN == WEB ESTATICA.
- CUIDADO DE NO VOLVER A LOS 90 :P

EASING



- NO CAMBIA TIMING, SOLO LA DISTRIBUCIÓN DE FRAMES EN EL ESPACIO.
- EASING CREA Y REFUERZA LA NATURALIDAD DE LA UX.
- CREA UN SENTIDO DE CONTINUIDAD CUANDO LOS OBJETOS SE COMPORTAN COMO LOS USUARIOS ESPERAN.

¿CUANDO USARLOS? SIEMPRE!
HTTP://GIZMA.COM/EASING

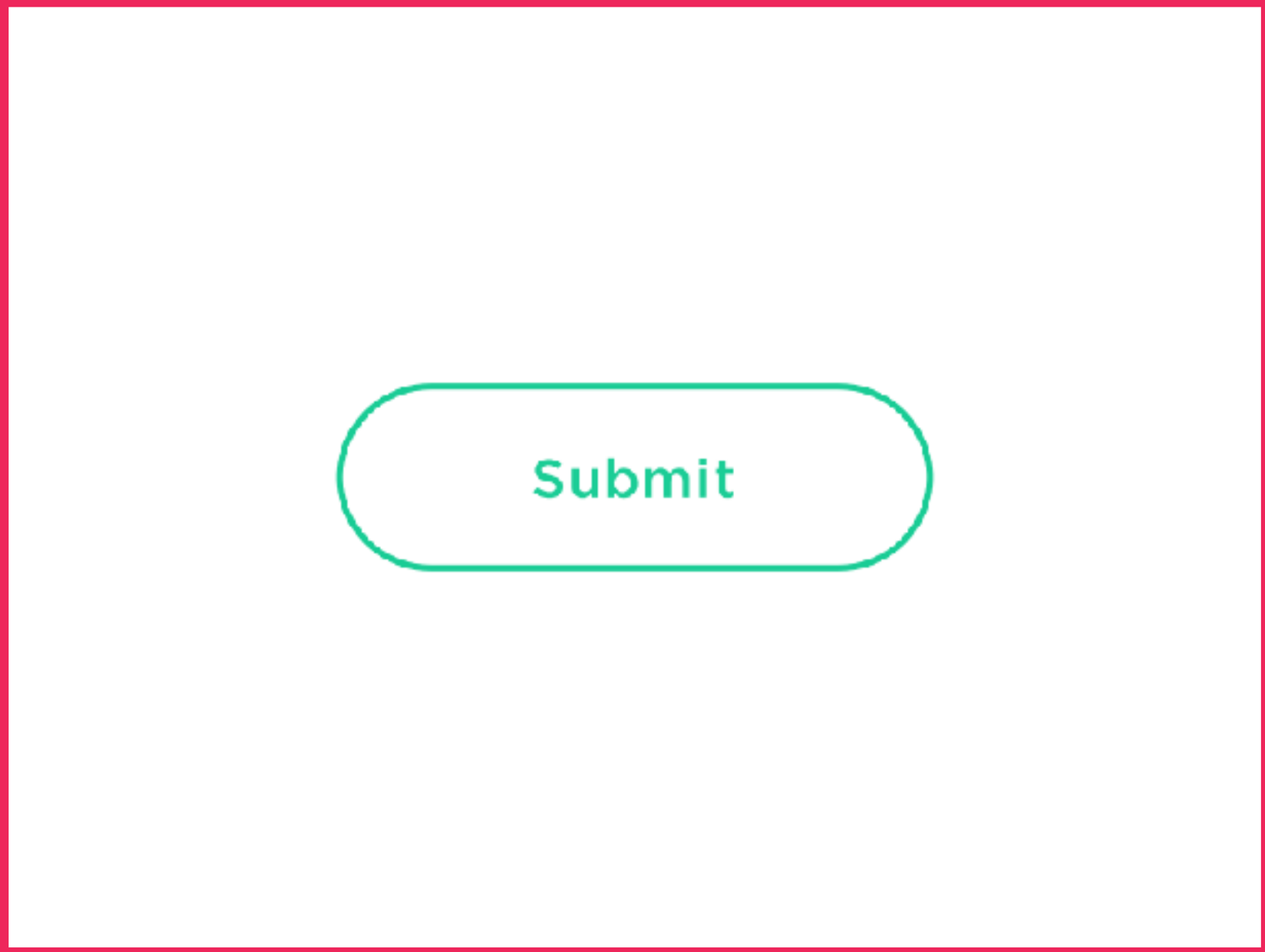
DELAY

CUANDO ANIMAMOS VARIOS ELEMENTOS A LA VEZ, ENTRE ELLOS TIENE QUE HABER UNA RELACIÓN. SI SE MOVIERAN TODOS A LA VEZ, NO SERÍA NADA NATURAL. TAMBIEN AYUDA A DETERMINAR QUE ELEMENTOS SON PRINCIPALES EN LA PANTALLA.

ANIMACIÓN SECUNDARIA

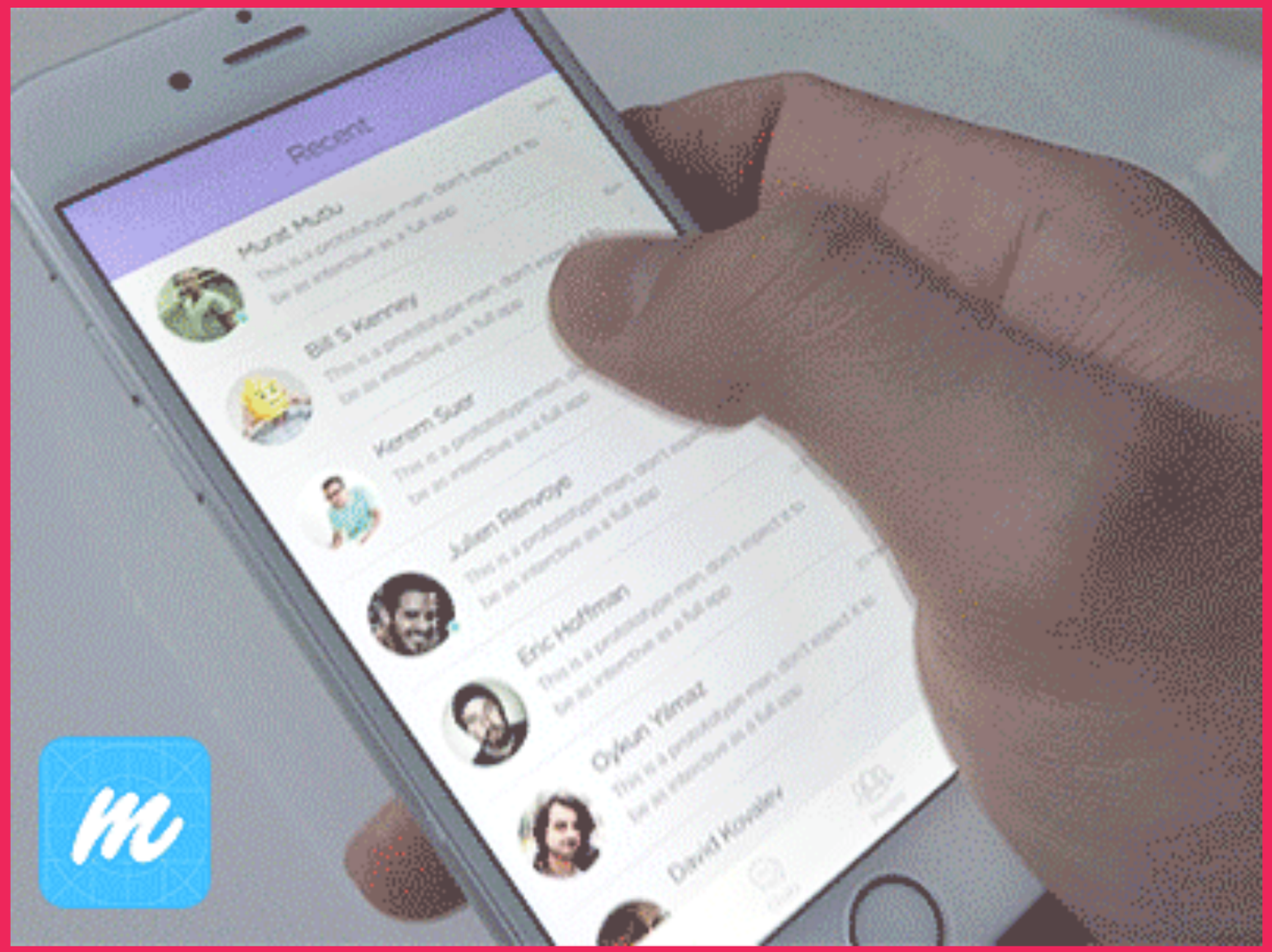
CUANDO EL USUARIO INTERACTÚA CON NUESTRA INTERFAZ Y PROVOCA CAMBIOS, ESTOS DEBEN REACCIONAR AL USUARIO E INFORMARLE DE LO QUE ESTÁ HACIENDO.

TRANSFORMACIÓN



PROCURAREMOS NO HACER DESAPARECER Y APARECER ELEMENTOS, INTENTADO PASAR DE UNOS A OTROS, TRANSFORMÁNDOLOS. EL USUARIO MANTENDRÁ LA MIRADA Y NO TENDRÁ QUE PROCESAR QUE OCURRE.

FOCALIZAR



CUANDO NECESITEMOS QUE EL USUARIO DECIDA ALGO Y NO QUEREMOS QUE PUEDA INTERACTUAR CON NADA MÁS.

QUE NO SEA UNA FERIA

ENLACES DE INTERÉS

- HTTPS://UXINMOTION.NET
- HTTP://ES.GIZMODO.COM/12-PRINCIPIOS-CLASICOS-DE-LA-ANIMACION-EN-12-GIFS-1570884994
- HTTPS://MATERIAL.IO/GUIDELINES/MOTION/MATERIAL-MOTION.HTML

CSS

TRANSITIONS

TRANSITION

- UNA TRANSICIÓN HACE QUE UN CAMBIO EN CSS SE HAGA POCO A POCO.

SIN TRANSITION (INSTANTANEO)

CON TRANSITION

TRANSITION

- PARA AÑADIR UNA TRANSICIÓN, AÑADIMOS LA PROPIEDAD TRANSITION.
- DEBEMOS PONER LA PROPIEDAD A ANIMAR Y EL TIEMPO DE LA ANIMACIÓN.

CSS

```
selector {
    transition: property <time>s;
  }
```

- POR EJEMPLO:

CSS

```
.btn {
    transition: background-color 0.5s;
    background: #4CAF50;
}
```

- OJO! PONER TRANSITION A LA CLASE GÉNERICA, NO AL OVER, SI NO SOLO LO HACE CUANDO SE APLICA LA CLASE SECUNDARIA O PSEUDO SELECTOR:

CSS

```
.btn:hover {
    transition: background-color 0.5s;
    background: #238027;
  }
```

EJERCICIO

TRANSITION

- EL EJEMPLO MÁS SENCILLO ES CUANDO HACEMOS HOVER EN UN BOTÓN
- EL USUARIO VA A RECIBIR FEEDBACK VISUAL, CAMBIANDO EL COLOR DE FONDO.

1 - HOVER TRANSITION INIT

HTML

```
<div class="login-page">
    <a class="btn btn-open" href='#'>LOGIN</a>
</div>
```

CSS

```
.btn {
    background: #4CAF50;
    width: 100%;
    . . .
  }

  .btn:hover {
    background: #238027;
  }
```

QUEREMOS QUE EL CAMBIO DE COLOR DE FONDO NO SEA INMEDIATO

CSS

```
.btn {
    transition: background-color 0.5s;
    background: #4CAF50;
}
```

1 - HOVER TRANSITION END

TRANSITION

- PODEMOS USAR ALL PARA TRANSICIONES DE TODAS LAS PROPERTIES
- PODEMOS AÑADIR TRANSITION-PROPERTY PARA ESPECIFICAR MÁS DE UNA.

CSS

```css
.selector {
    transition: all 1s;
    transition-property: property1, property2, . . .;
}
```

TRANSITION

EJERCICIO

2 - HOVER TRANSITION MULTIPLE INIT

- PARA NUESTRO EJEMPLO, AÑADIMOS DOS SPAN CON TEXTO DIFERENTE Y HACEMOS QUE CAMBIE DE UNO A OTRO:

HTML

```
<a class="btn btn-open" href='#'>
    <span class='top content'>OPEN LOGIN</span>
    <span class='bottom content'>LOG NOW!</span>
</a>
```

EJERCICIO

TRANSITION

- CON CSS HACEMOS QUE SE OCULTE Y SE MUESTRE EN EL HOVER:

CSS

```css
btn .content {
  position: absolute;
  display: block;
  width: 100%;
  padding-top: 14px;
}

.top.content { top: 0}
.bottom.content { top: 50px}
.btn:hover .top { top: -50px}
.btn:hover .bottom { top: 0}
```

2 - HOVER TRANSITION MULTIPLE MED

- FINALMENTE METEMOS NUESTRA ANIMACIÓN:

CSS

```css
.btn, .btn .content {
    transition:all .5s;
}
```

2 - HOVER TRANSITION MULTIPLE END

TRANSITION

- ADEMAS, SI QUEREMOS PODEMOS AÑADIR TANTO UNA FUNCIÓN DE EASE, Y UN DELAY.

HTTP://EASINGS.NET/

CSS

```
selector {
    transition: property <time>s <ease> <delay>s;
  }
```

- PODEMOS CREAR NUESTRAS PROPIAS TRANSICIONES:

HTTP://CUBIC-BEZIER.COM/

EJERCICIO

TRANSITION

- EN EL EJEMPLO, CREAMOS UNA FUNCIÓN QUE TENGA UN REBOTE ANTES Y DESPUES:

HTTP://CUBIC-BEZIER.COM/#.5,-0.99,.5,2

- SEPARAMOS LAS ANIMACIONES DE BCN Y CONTENT Y AÑADIMOS ANTICIPACIÓN CON UNA FUNCIÓN BEZIER, Y DELAY:

CSS

```css
.btn {
    transition:all .5s ease-in-out;
  }

.btn .content {
    transition:all .5s cubic-bezier(.5,-1,.5,2) 0.3s;
  }
```

2 - HOVER TRANSITION MULTIPLE END EASE

EJERCICIO

TRANSITION

- EMPEZAMOS CON UN FORMULARIO DE LOGIN

3 - LOGIN FORM – PLACEHOLDERS INIT

- LO QUE QUEREMOS ES HACER ALGO SIMILAR A LOS FORMULARIOS DE ANDROID - MATERIAL DESIGN

EJEMPLO

- LO PRIMERO ES COLOCAR EL LABEL COMO SI DE UN PLACEHOLDER SE TRATARA:

CSS

```css
label {
    color: #777;
    position: relative;
    top: 30px;
    padding: 0 14px;
  }
```

3 - LOGIN FORM – PLACEHOLDERS - MED

EJERCICIO

TRANSITION

- AL FINAL, VAMOS A SELECCIONAR POR CSS LOS LABEL QUE TENGAN UN INPUT EN FOCUS O RELLENO, Y PARA NO TENER QUE USAR JS, TENEMOS QUE USAR SELECTORES ESPECIALES DE CSS:

W3SCHOOL CSS SELECTORS

- CAMBIAMOS EL ORDEN DEL HTML, DEJANDO EL LABEL DETRÁS DEL INPUT, Y MOVIÉNDOLO POR CSS:

HTML

```
<input type="text" id='user' required/>
<label for='user'>username</label>
...
<input type="password" id="password" required/>
<label for='password'>password</label>
```

CSS

```
label {
    color: #777;
    position: relative;
    padding: 0 14px;
    top: -31px;
    pointer-events: none;
}
```

TRANSITION

EJERCICIO

- Y FINALMENTE METEMOS LAS ANIMACIONES:

CSS

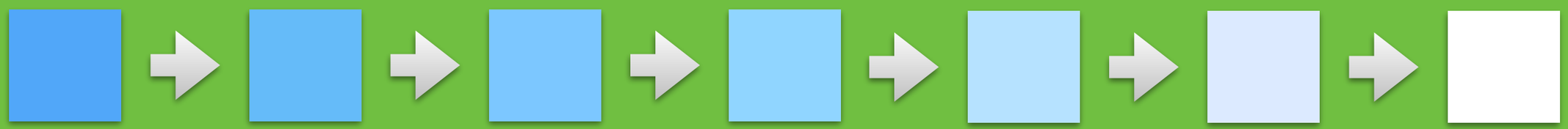
```css
label {
    color: #777;
    position: relative;
    padding: 0 14px;
    top: -31px;
    pointer-events: none;
    transition: all 0.5s;
  }

  input:focus + label, input:valid + label {
      transform: translateY(-32px) translateX(-23px) scale(0.9);
  }
```

3 - LOGIN FORM – PLACEHOLDERS - END

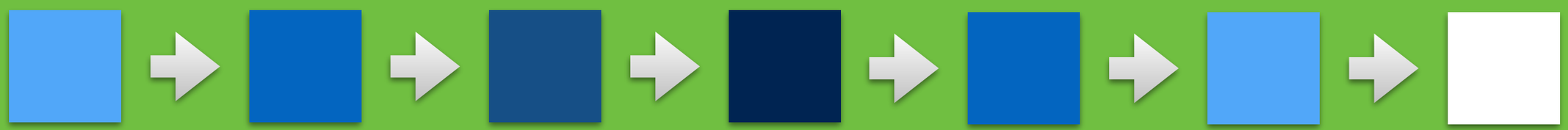
KEYFRAMES & ANIMATION

- UNA ANIMACIÓN ES UN GRUPO DE KEYFRAMES CON TRANSICIONES ENTRE ELLOS
- UN KEYFRAME ES UN ESTADO CONCRETO DE LA ANIMACIÓN. EN UNA TRANSICIÓN DEFINIMOS DOS ESTADOS, INICIAL Y FINAL. CON LOS KEYFRAMES DEFINIMOS ESTADOS INTERMEDIOS.
- ENTRE KEYFRAMES SE TRANSICIONA IGUAL QUE HASTA AHORA.

SIN KEYFRAMES:



CON KEYFRAMES:



KEYFRAME

KEYFRAMES & ANIMATION

- @KEYFRAMES SE USA PARA DEFINIR POR LOS ESTADOS QUE QUEREMOS PASAR
- EN CADA LINEA SE DEFINE EL % DE LA ANIMACIÓN EN LA QUE SE APLICA ESE ESTADO, Y LAS PROPIEDADES CSS QUE APLICAN
- A PARTIR DE AHORA A CADA LINEA LA LLAMAREMOS KEYFRAME

CSS

```
@keyframes name {
  xx%  {properties}
  ...
  xx%  {properties}
}
```

- POR EJEMPLO:

CSS

```
@keyframes anim2 {
  0%   {top: 0px; left: 0px; background: red;}
  25%  {top: 0px; left: 100px; background: blue;}
  50%  {top: 100px; left: 100px; background: yellow;}
  75%  {top: 100px; left: 0px; background: green;}
  100% {top: 0px; left: 0px; background: red;}
}
```

KEYFRAMES & ANIMATION

- ANIMATION TIENE MUCHAS PROPIEDADES:

  W3SCHOOL ANIMATION

- NOSOTROS VAMOS A USAR SOLO LAS PRINCIPALES:

CSS

```
selector {
    animation: keyframesname <time>s <repeat>;
}
```

- POR EJEMPLO:

CSS

```
.btn {
  animation: anima1 0.5s;
}

.btn2 {
  animation: anima2 0.5s infinite;
}
```

EJERCICIO

KEYFRAMES & ANIMATION

- EMPEZAMOS CON NUESTRO FORMULARIO DE LOGIN

4 - LOGIN FORM - WITH ERRORS - INIT

- QUEREMOS "SACUDIR" EL PASSWORD SI NO ES CORRECTO.

- EMPEZAMOS APLICANDO EL BORDE ROJO Y UN ANIMATION DE UN GRUPO DE KEYFRAMES QUE CREAREMOS DESPUES:

CSS

```css
input.withErrors {
  animation: shake 0.8s;
  border-color: tomato;
}
```

EJERCICIO

KEYFRAMES & ANIMATION

- CREAMOS LOS KEYFRAMES QUE SACUDEN EL INPUT

CSS

```css
@keyframes shake {
  10% { transform: translate3d(-2px, 0, 0); }
  20% { transform: translate3d( 2px, 0, 0); }
  30% { transform: translate3d(-4px, 0, 0); }
  40% { transform: translate3d( 4px, 0, 0); }
  50% { transform: translate3d(-4px, 0, 0); }
  60% { transform: translate3d( 4px, 0, 0); }
  70% { transform: translate3d(-2px, 0, 0); }
  80% { transform: translate3d( 2px, 0, 0); }
  90% { transform: translate3d(-2px, 0, 0); }
}
```

- PODEMOS AGRUPARLO DE ESTA MANERA:

CSS

```css
@keyframes shake {
  10%, 70%, 90% { transform: translate3d(-2px, 0, 0); }
  20%, 80%      { transform: translate3d( 2px, 0, 0); }
  30%, 50%      { transform: translate3d(-4px, 0, 0); }
  40%, 60%      { transform: translate3d( 4px, 0, 0); }
}
```

4 - LOGIN FORM - WITH ERRORS - COMPLETE

EJERCICIO

KEYFRAMES & ANIMATION

- CREAMOS LOS KEYFRAMES QUE SACUDEN EL INPUT

CSS

```css
@keyframes shake {
  10% { transform: translate3d(-2px, 0, 0); }
  20% { transform: translate3d( 2px, 0, 0); }
  30% { transform: translate3d(-4px, 0, 0); }
  40% { transform: translate3d( 4px, 0, 0); }
  50% { transform: translate3d(-4px, 0, 0); }
  60% { transform: translate3d( 4px, 0, 0); }
  70% { transform: translate3d(-2px, 0, 0); }
  80% { transform: translate3d( 2px, 0, 0); }
  90% { transform: translate3d(-2px, 0, 0); }
}
```

- PODEMOS AGRUPARLO DE ESTA MANERA:

CSS

```css
@keyframes shake {
  10%, 70%, 90% { transform: translate3d(-2px, 0, 0); }
  20%, 80%      { transform: translate3d( 2px, 0, 0); }
  30%, 50%      { transform: translate3d(-4px, 0, 0); }
  40%, 60%      { transform: translate3d( 4px, 0, 0); }
}
```

4 - LOGIN FORM - WITH ERRORS - COMPLETE

EJERCICIO

KEYFRAMES & ANIMATION

- VAMOS A JUNTAR LOS DOS EJERCICIOS, AL HACER CLICK EN EL BOTON, CUBRIMOS CON UNA CAPA TODO Y ABRIMOS EL FORMULARIO DE LOGIN.

5 - OPEN LOGIN FORM - INIT

- HACEMOS QUE LA CAPA DE EN MEDIO TENGA 1 PIXEL DE ALTO Y 0 DE ANCHO
- CREAMOS LOS KEYFRAMES PARA QUE ENSANCHE Y LUEGO ESTIRE
- APLICAMOS CON ANIMATION LA ANIMACIÓN.

CSS

```css
.layer {
  transform: scaleY(0.01) scaleX(0);
}

@keyframes unfoldIn {
  0%   { transform: scaleY(.005) scaleX(0); }
  50%  { transform: scaleY(.005) scaleX(1); }
  100% { transform: scaleY(1) scaleX(1); }
}

.layer.opened {
  animation: unfoldIn 1s;
  transform: scaleY(1) scaleX(1);
}
```

EJERCICIO

KEYFRAMES & ANIMATION

- HACEMOS LO MISMO CON EL FORMULARIO. SKEW NOS INCLINA X GRADOS UN ELEMENTO A LA DERECHA.
- AUMENTAMOS Y DISMINUIMOS LA ESCALA EN X PARA IR HACIENDO TODO MÁS SUAVE (PRINCIPIO SKETCH & SKEW)

CSS

```css
.form {
  display: none;
  transform:translateX(-1500px);
}

@keyframes roadRunnerIn {
  0%   { transform:translateX(-1500px) skewX(30deg) scaleX(1.3); }
  50%  { transform:translateX(-1500px) skewX(30deg) scaleX(1.3); }
  90%  { transform:translateX(30px)    skewX(0deg)  scaleX(.9);  }
  100% { transform:translateX(0px)     skewX(0deg)  scaleX(1);   }
}

.form.opened {
  animation: roadRunnerIn 2s;
  transform:translateX(0px);
  display:inherit;
}
```

5 - OPEN LOGIN FORM - COMPLETE

JS

VANILLA

EJERCICIO

EASE A MANO

- A VECES CON ANIMACIONES CSS NO TENEMOS SUFICIENTE, O BIEN POR QUE LAS ANIMACIONES SON DINÁMICAS, PORQUE QUEREMOS CONTROLARLAS, SINCRONIZARLAS, ETC.

- VAMOS A HACER UN EJERCICIO MUY SIMPLE PARA ENTENDER COMO HACER A MANO UNA ANIMACIÓN CON JS Y VER COMO FUNCIONA EL EASE.

EASING JS BASICO INIT

- EMPEZAMOS CON UN PELOTITA ROJA QUE AL HACER CLICK EN EL BODY VA A EJECUTAR UNA FUNCIÓN QUE LA SE VA A POSICIONA DONDE HAGAMOS CLICK.

EJERCICIO

EASE A MANO

- CREAMOS DOS OBJETOS CON PROPIEDADES X, Y. UNA NOS VA A GUARDAR LA POSICIÓN ACTUAL, OTRA LA POSICIÓN DE DESTINO DE LA ANIMACIÓN.

JS

```
var temp = {x:0, y:0},
    destino = {x:0, y:0};
```

- CAMBIAMOS EL CLICK PARA QUE CAMBIE LOS VALORES DE DESTINO Y EJECUTAMOS LA FUNCIÓN SETPOSITION:

JS

```
$('document').ready(function() {
    $('body').click(function (event) {
        destino.x = event.pageX;
        destino.y = event.pageY;
    });
    setPosition();
});
```

EASE A MANO

- ESTA SENCILLA FUNCIÓN NOS SIRVE TANTO PARA ANIMAR COMO PARA EL EASE.

```js
pos = pos + (destino - pos)/10;
```

- LO QUE ESTAMOS HACIENDO ES SUMAR A LA POSICIÓN ACTUAL UNA DÉCIMA PARTE DE LO QUE NOS QUEDA PARA LLEGAR.

- SI POR EJEMPLO ESTAMOS EN X=0 Y EL DESTINO ES X=10, EN EL PRIMER FRAME X SERA X + (10 - 0) / 10 = 1. EN EL SIGUIENTE FRAME, COMO ESTAMOS EN X=1, (10-1) / 10 = 0.9, CON LO QUE CADA VEZ AVANZAMOS MAS DESPACIO, DANDO ESA SENSACIÓN DE DESACELERACIÓN. VEÁMOSLOS EN UNA TABLA:

Frame	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Posición	0	1	1.9	2.71	3.44	4	4.6	5.14	5.89	6.3
Animación	1	0.9	0.81	0.73	0.66	0.6	0.54	0.49	0.41	0.37

- SI EN VEZ DE 10 PONEMOS 2, LA ANIMACIÓN SERÁ MÁS RÁPIDA Y CON MENOS SUAVIDAD.

- SI PONEMOS 20, SERÁ MÁS LENTA Y CON MÁS SUAVIDAD.

EJERCICIO

EASE A MANO

- ELIMINAMOS LOS PARAMETROS DE SETPOSITION Y MODIFICAMOS LA POSICIÓN TEMP CON LA FUNCIÓN DE EASE:

JS

```
temp.x += (destino.x - temp.x) / 10;
temp.y += (destino.y - temp.y) / 10;
```

- SETEAMOS LA POSICIÓN DE LA BOLITA CON EL OBJETO TEMP

JS

```
$('#follow').css({left:temp.x-10+'px', top:temp.y-10+'px'});
window.requestAnimationFrame(setPosition);
```

- Y EJECUTAMOS SET ANIMATION FRAME, QUE HARÁ QUE SE EJECUTE DE FORMA RECURSIVA LA FUNCIÓN SETPOSITION CADA FRAME QUE PINTE EL NAVEGADOR.

RESUMEN

- NO TODAS LAS PROPIEDADES PUEDEN SER TRANSICIONABLES.
- EN GENERAL LAS NUMÉRICAS SI Y LAS NO NUMÉRICAS NO.
- HAY EXCEPCIONES, POR EJEMPLO DISPLAY NO ES TRANSICIONABLE, PERO SI SE PUEDEN TRANSICIONAR VISIBILITY, QUE OCULTARÁ EL OBJETO AL FINALIZAR LA TRANSICIÓN. MUY UTIL CUANDO SE QUIERE ANIMAR LA DESAPARICIÓN DE UN ELEMENTO.

- LISTADO DE PROPIEDADES: HTTPS://DEVELOPER.MOZILLA.ORG/EN-US/DOCS/WEB/CSS/CSS_ANIMATED_PROPERTIES

- FRAMEWORKS DE ANIMACIONES SIMPLES (CLASES QUE APLICÁNDOLAS HACEN ANIMACIONES SIMPLE)

  - **Animate.css**
  - **Bounce.js**
  - **AnimeJS**
  - **Magic Animations**
  - **DynCSS**
  - **CSShake**
  - **Hover.CSS**
  - **Velocity.js**
  - **AniJS**