Esqueleto HTML

La estructura básica de nuestro fichero principal será la siguiente:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
<link href="calculator.css" rel="stylesheet">
<script src="calculator.js"></script>
<title>Calculator!</title>
</head>
<body>
...
</body>
</html>
```

HTML5 doctype

La primera línea que aparece en el archivo (

```
<!doctype html>
```

) no es tanto una etiqueta HTML, sino una declaración del lenguaje y la versión que vamos a utilizar (HTML 5). Además, como no se trata de una etiqueta, no necesitamos cerrar la declaración:

```
<!doctype html>
<html lang="en">
...
</html>
```

Etiqueta para obtener una visualización *responsive*

El *viewport* es el área de una página web que visualiza un usuario. Puede variar dependiendo del dispositivo, y será menor por ejemplo en un teléfono móvil que en la pantalla de un ordenador.

Antes de que aparecieran las tabletas y los teléfonos móviles, las páginas web se diseñaban sólo para pantallas de ordenadores, y era muy común que tuvieran un diseño estático y un tamaño fijo.

Cuando comenzamos a utilizar los navegadores de los dispositivos móviles, las páginas de tamaño fijo eran demasiado grandes para ajustarse al *viewport*. Para solucionar esto, los navegadores modernos reducían el tamaño de todos los elementos de la página web para ajustarse a la pantalla, con lo que resultaba muy difícil leer el contenido de la misma.

Un elemento

<**meta**>
de tipo *viewport* le dice al navegador cómo controlar las dimensiones de la página para hacer zoom automáticamente de forma adecuada, utilizando por ejemplo un mayor tamaño de letra y habilitando a su vez scroll vertical para poder acceder a todo el contenido.
En resumen, para asegurar una correcta visualización, tanto en dispositivos móviles como en ordenadores de escritorio, debemos añadir una etiqueta

<**meta name**="viewport">
para especificar el tamaño y la escala del contenido de la pantalla:

<**meta name**="viewport" **content**="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">

Más concretamente, con la cadena especificada dentro del atributo *content* del ejemplo, estaríamos diciéndole al *WebView* (el navegador interno del dispositivo) que aproveche al máximo el área visible de la pantalla. Y puesto que la visualización debería ser óptima, el usuario no necesitará realizar zoom (

user-scalable=no
).
Puedes encontrar más información aquí. Además, también puedes consultar algunas recomendaciones al respecto de los desarrolladores de Ionic, o de Safari o de Firefox. Además, debido a la constante actualización en el mercado de dispositivos móviles puede resultar necesario realizar algún ajuste, como por ejemplo, para obtener una mejor visualización en el iPhone X.
También es interesante observar la cadena de configuración utilizada por algunas librerías, como por ejemplo Bootstrap.

Colocando los botones en una tabla

Para distribuir los botones de la calculadora en varias filas y columnas utilizaremos una tabla. Por ejemplo, para obtener la siguiente distribución:

0			
1	2	3	+
4	5	6	–
7	8	9	*
0			/
AC		.	=

Podemos utilizar el siguiente código:

```
<table>
<tr>
<td colspan="4">0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>-</td>
```

```
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>*</td>
</tr>
<tr>
<td colspan="3">0</td>
<td>/</td>
</tr>
<tr>
<td colspan="2">AC</td>
<td>.</td>
<td>=</td>
</tr>
</table>
```

Como se puede observar, se utiliza el elemento

<tr></tr>
(*table row*) para establecer las filas de la tabla, y el elemento
<td></td>
(*table data*) para delimitar las celdas de cada fila.
Añadimos el atributo

colspan="2"
y
colspan="3"
para hacer que una determinada celda abarque 2 y 3 columnas respectivamente, y por lo tanto, el botón que contiene sea más grande que el resto.

Botones básicos para capturar las pulsaciones

Utilizaremos el elemento

<button></button>
para implementar las teclas de la calculadora, y los colocaremos dentro de cada celda de la tabla. Por ejemplo:

```
<tr>
<td><button onclick="...">1</button></td>
<td><button onclick="...">2</button></td>
<td><button onclick="...">3</button></td>
<td><button onclick="...">+</button></td>
</tr>
```

El atributo *onclick* nos servirá para ejecutar una función de JavaScript cada vez que el usuario pulse ese botón. Lo veremos con detalle un poco más adelante.

Utilizando archivos CSS y JavaScript externos

Para poder cambiar el aspecto de la calculadora, utilizaremos un fichero externo que contendrá algunas líneas de código CSS en un fichero externo, que incluiremos en el archivo principal (**index.html**).

Además, también utilizaremos diversas funciones y código JavaScript que también ubicaremos en un archivo externo.

Para incluir ambos archivos en el código HTML principal y poder utilizar el código que hayamos colocado allí, utilizaremos las siguientes etiquetas:

```
<link href="calculator.css" rel="stylesheet">
<script src="calculator.js"></script>
```

Modificando el aspecto de los botones

Para conseguir que los botones sean un poco más grandes y se puedan pulsar más fácilmente, añadiremos algunas propiedades CSS al archivo **calculator.css**:

```
table {
width: 100%;
}
button {
width: 100%;
font-size: 150%;
}
.ac {
color: red;
}
```

Mediante la propiedad

```
width: 100%;
```

haremos que la tabla ocupe todo el ancho de la pantalla, y que los botones ocupen todo el ancho de cada celda de la tabla.
Con la modificación

```
font-size: 150%;
```

incrementaremos el tamaño del texto que pongamos dentro de los botones.
Mediante la definición de una clase, conseguiremos aplicar un estilo concreto (color del texto en rojo) al botón **AC** (*All Clear*):

```
.ac {
color: red;
}
```

```
<button class="ac">AC</button>
```

Añadiendo la funcionalidad para realizar los cálculos

El código HTML ya incluye todos los elementos básicos para interactuar con el usuario, pero todavía no disponemos del código JavaScript necesario para capturar las pulsaciones de cada tecla, actualizar la pantalla de la calculadora y calcular el resultado.

Proponemos utilizar cinco funciones, cuyo propósito justificamos debajo del cuadro:

```
function setResult(value) {
document.getElementById('result').innerHTML = value;
}
function getResult() {
return(document.getElementById('result').innerHTML);
}
function add(key) {
var result = getResult();
if (result!='0' || isNaN(key)) setResult(result + key);
else setResult(key);
}
function calc() {
var result = eval(getResult());
setResult(result);
}
function del() {
setResult(0);
}
```

- *setResult()*: Actualiza la pantalla de la calculadora poniendo el valor que se pase como parámetro.
- *getResult()*: Recoge el valor del último resultado obtenido, o de la expresión matemática que se debe calcular, y que se está visualizando en la pantalla de la calculadora.
- *add()*: Añade a la pantalla la tecla pulsada (por ejemplo, el dígito o la operación a realizar). Si la pantalla ya contiene algún dato o la tecla que se pulsa no es un dígito, el carácter de la tecla pulsada se añadirá a la pantalla. En caso contrario (por ejemplo, si la pantalla está a cero, y se pulsa otro dígito), el contenido de la pantalla se reemplazará con la tecla pulsada.
- *calc()*: Realiza el cálculo de la expresión que se encuentre en la pantalla (utilizando la función

  `eval()`

  ), y escribe el resultado.

- *del()*: Pone a cero el contenido de la pantalla de la calculadora.

  Las tres últimas funciones de la lista se ejecutarán desde el código HTML utilizando el atributo *onclick*:

```
<button onclick="add('1')">1</button>
...
<button onclick="calc()">=</button>
...
<button onclick="del()">AC</button>
```

En resumen...

El fichero *index.html*

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<meta name="viewport" content="viewport-fit=cover, width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
<link href="calculator.css" rel="stylesheet">
<script src="calculator.js"></script>
<title>Calculator!</title>
</head>
<body>
<table>
<tr>
<td colspan="4"><button id="result">0</button></td>
</tr>
<tr>
<td><button onclick="add('1')">1</button></td>
<td><button onclick="add('2')">2</button></td>
<td><button onclick="add('3')">3</button></td>
<td><button onclick="add('+')">+</button></td>
</tr>
<tr>
<td><button onclick="add('4')">4</button></td>
<td><button onclick="add('5')">5</button></td>
<td><button onclick="add('6')">6</button></td>
<td><button onclick="add('-')">-</button></td>
</tr>
<tr>
<td><button onclick="add('7')">7</button></td>
<td><button onclick="add('8')">8</button></td>
```

```
<td><button onclick="add('9')">9</button></td>
<td><button onclick="add('*')">*</button></td>
</tr>
<tr>
<td colspan="3"><button onclick="add('0')">0</button></td>
<td colspan="3"><button onclick="add('/')">/</button></td>
</tr>
<tr>
<td colspan="2"><button onclick="del()" class="ac">AC</button></td>
<td><button onclick="add('.')">.</button></td>
<td><button onclick="calc()">=</button></td>
</tr>
</table>
</body>
</html>
```

El fichero *calculator.css*

```
table {
width: 100%;
}
button {
width: 100%;
font-size: 150%;
}
.ac {
color: red;
}
```

El fichero *calculator.js*

```
function setResult(value) {
document.getElementById('result').innerHTML = value;
}
function getResult() {
return(document.getElementById('result').innerHTML);
}
function add(key) {
var result = getResult();
if (result!='0' || isNaN(key)) setResult(result + key);
else setResult(key);
}
function calc() {
var result = eval(getResult());
setResult(result);
}
function del() {
setResult(0);
}
```