

Promise

El objeto **promise** o promesa representa un valor que puede estar disponible ahora, en el futuro o nunca. (¡igual que una promesa en la vida real!). Se utiliza cuando estamos frente a un código asíncrono. Ya que, valga la redundancia, este objeto “nos promete” que devolverá un valor en una línea de tiempo presente o futura y *recuerda* el contexto en donde se ejecuta, es decir, sabe con precisión en qué punto se ha de resolver un valor o lanzar un error.

Para la utilización de promesas vamos a poder utilizare diferentes formas, vamos a empezar viendo una primera forma, que nos resulte fácil de entender.

Para crear una promesa deberemos poner
variable= new Promise(nombre-función-1)

Creamos una promesa e indicamos el nombre de una función, que contiene el código que se va a ejecutar cuando se realiza la promesa (la acción que prometemos que vamos a desarrollar).

ejemplo, creamos una promesa, que cuando se realice se va a ejecutar la función tratar

| | |
|-----|---|
| 001 | let mipromesa= new Promise(tratar); |
|-----|---|

Crearemos una promesa en un función y dicha función devolver la promesa.

```
function crearPromesa(){
    let variable= new Promise(nombre-función-1);
    return variable
}
```

ejemplo, tenemos la función crear que crea una promesa, que cuando se realice se va a ejecutar la función tratar y que la función crear devuelve la promesa que hemos creado.

| | |
|-----|---|
| 001 | function crear(){ |
| 002 | let mipromesa= new Promise(tratar); |
| 003 | return mipromesa; |
| 004 | } |

Deberemos crearnos una función que va a ser la encargada de prometer la promera, esto es, indicar que se realice la promesa.

```
function nombre-función(){
    crearPromesa().then(función-resuelve).catch(función-error);
}
```

O bien

```
function nombre-función(){
    crearPromesa()
        .then(función-resuelve)
        .catch(función-error);
}
```

En el método **then** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones prometidas.

En el método **catch** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones que se van a realizar cuando se rechaza la promesa o hay un error en la misma.

| | |
|-----|----------------------------|
| 001 | function enviar() { |
| 002 | crear() |
| 003 | .then(correcto) |
| 004 | .catch(errores) |
| 005 | } |

Vamos a declarar la función que se va a ejecutar cuando se realiza la promesa, lo que prometemos que vamos a hacer.

```
function nombre-función-1(función-resolver, función-rechazar){
    cuerpo de la función.
}
```

Esta función va a recibir dos parámetros:

- El primer parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se acepta la promesa, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de resolución de la promesa.
- El segundo parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se rechaza la promesa o cuando existe un error, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de rechazo o error de la promesa.

En el cuerpo de la función vamos a indicar que deseamos hacer y en un momento llamaremos a la función que resuelve la promesa, en este caso se la puede pasar varios parámetros, y en otro momento llamaremos a la función que rechaza la promesa.

Ejemplo: en este caso cogemos el valor de dos cajas de texto, si el valor de ambas es distinto de la cadena vacía llamamos a la función resolver con el valor de las dos cajas de texto y si no se llama a la función rechazar.

| | |
|-----|---|
| 001 | function tratar(resolver, rechazar) { |
| 002 | let nom=document.getElementById("nombre").value.trim(); |
| 003 | let ape=document.getElementById("apellidos").value.trim(); |
| 004 | if (nom != "" && ape != "") |
| 005 | resolver(nom , ape) |
| 006 | else |
| 007 | rechazar(); |
| 008 | } |

A continuación nos declaramos la función que resuelve la promesa

```
function función-resuelve([parametros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto la concatenación de los dos parámetros

| | |
|-----|--|
| 001 | function correcto(unos,dos) { |
| 002 | document.getElementById("resultado").value=unos + " " + dos; |
| 003 | } |

Y por último nos declaramos la función de rechazo de la promesa o de error de la promesa

```
function función-error([parámetros]){
    cuerpo-función
}
```

}

Ejemplo: mostramos en una caja de texto un mensaje de error.

| | |
|-----|--|
| 001 | <code>function errores() {</code> |
| 002 | <code> document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 003 | <code>}</code> |

Ahora vamos a ver el programa completo

promise101.js

| | |
|-----|--|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code> window.addEventListener("load", inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code> window.attachEvent("onload", inicio);</code> |
| 005 | <code>function inicio() {</code> |
| 006 | <code> let boton=document.getElementById("poner");</code> |
| 007 | <code> if (document.addEventListener)</code> |
| 008 | <code> boton.addEventListener("click", enviar)</code> |
| 009 | <code> else if (document.attachEvent)</code> |
| 010 | <code> boton.attachEvent("onclick", enviar);</code> |
| 011 | <code>}</code> |
| 012 | <code>function enviar() {</code> |
| 013 | <code> crear()</code> |
| 014 | <code> .then(correcto)</code> |
| 015 | <code> .catch(errores)</code> |
| 016 | <code>}</code> |
| 017 | <code>function crear() {</code> |
| 018 | <code> let mipromesa= new Promise(tratar);</code> |
| 019 | <code> return mipromesa;</code> |
| 020 | <code>}</code> |
| 021 | <code>function tratar(resolver, rechazar){</code> |
| 022 | <code> let nom=document.getElementById("nombre").value.trim();</code> |
| 023 | <code> let ape=document.getElementById("apellidos").value.trim();</code> |
| 024 | <code> if (nom!="" && ape!="")</code> |
| 025 | <code> resolver(nom,ape)</code> |
| 026 | <code> else</code> |
| 027 | <code> rechazar();</code> |
| 028 | <code>}</code> |
| 029 | <code>function correcto(unos,dos){</code> |
| 030 | <code> document.getElementById("resultado").value=unos + " " + dos;</code> |
| 031 | <code>}</code> |
| 032 | <code>function errores() {</code> |
| 033 | <code> document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 034 | <code>}</code> |

Vamos a ver varios ejemplos del mismo programa haciéndolo de otra forma

promise102.js

| | |
|-----|---|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code> window.addEventListener("load", inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code> window.attachEvent("onload", inicio);</code> |
| 005 | <code>function inicio() {</code> |
| 006 | <code> let boton=document.getElementById("poner");</code> |
| 007 | <code> if (document.addEventListener)</code> |
| 008 | <code> boton.addEventListener("click", enviar)</code> |
| 009 | <code> else if (document.attachEvent)</code> |
| 010 | <code> boton.attachEvent("onclick", enviar);</code> |
| 011 | <code>}</code> |
| 012 | <code>function enviar() {</code> |
| 013 | <code> crear()</code> |
| 014 | <code> .then(correcto)</code> |
| 015 | <code> .catch(errores)</code> |
| 016 | <code>}</code> |
| 017 | <code>function crear() {</code> |
| 018 | <code> let mipromesa= new Promise(</code> |
| 019 | <code> function (resolver, rechazar){</code> |
| 020 | <code> let nom=document.getElementById("nombre").value.trim();</code> |
| 021 | <code> let ape=document.getElementById("apellidos").value.trim();</code> |
| 022 | <code> if (nom!="" && ape!="")</code> |

| | |
|-----|---|
| 023 | resolver(nom+ " " +ape) // solo se puede enviar un parámetro |
| 024 | else |
| 025 | rechazar(); |
| 026 | } |
| 027 |); |
| 028 | return mipromesa; |
| 029 | } |
| 030 | function correcto(unos){ |
| 031 | document.getElementById("resultado").value=unos; |
| 032 | } |
| 033 | function errores(){ |
| 034 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 035 | } |

promise103.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", enviar) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then(function (uno){ |
| 015 | document.getElementById("resultado").value=uno; |
| 016 | }) |
| 017 | .catch(function (){ |
| 018 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 019 | }) |
| 020 | } |
| 021 | function crear(){ |
| 022 | let mipromesa= new Promise(|
| 023 | function (resolver, rechazar){ |
| 024 | let nom=document.getElementById("nombre").value.trim(); |
| 025 | let ape=document.getElementById("apellidos").value.trim(); |
| 026 | if (nom!=" " && ape!=" ") |
| 027 | resolver(nom+ " " +ape) // solo se puede enviar un parámetro |
| 028 | else |
| 029 | rechazar(); |
| 030 | } |
| 031 |); |
| 032 | return mipromesa; |
| 033 | } |

promise104.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", enviar) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then((uno)=>{ |
| 015 | document.getElementById("resultado").value=uno; |
| 016 | }) |
| 017 | .catch(()=>{ |
| 018 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 019 | }) |

| | |
|-----|--|
| 020 | } |
| 021 | function crear(){ |
| 022 | let mipromesa= new Promise(|
| 023 | (resolver, rechazar) => { |
| 024 | let nom=document.getElementById("nombre").value.trim(); |
| 025 | let ape=document.getElementById("apellidos").value.trim(); |
| 026 | if (nom!=" " && ape!=" ") |
| 027 | resolver(nom+ " " +ape) // solo se puede enviar un parámetro |
| 028 | else |
| 029 | rechazar(); |
| 030 | } |
| 031 |); |
| 032 | return mipromesa; |
| 033 | } |

promise105.js

| | |
|-----|--|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load",inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload",inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click",enviar) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick",enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then(uno=>{ |
| 015 | document.getElementById("resultado").value=uno; |
| 016 | }) |
| 017 | .catch(() =>{ |
| 018 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 019 | }) |
| 020 | } |
| 021 | function crear(){ |
| 022 | let mipromesa= new Promise(|
| 023 | (resolver, rechazar) => { |
| 024 | let nom=document.getElementById("nombre").value.trim(); |
| 025 | let ape=document.getElementById("apellidos").value.trim(); |
| 026 | if (nom!=" " && ape!=" ") |
| 027 | resolver(nom+ " " +ape) // solo se puede enviar un parámetro |
| 028 | else |
| 029 | rechazar(); |
| 030 | } |
| 031 |); |
| 032 | return mipromesa; |
| 033 | } |

Una segunda forma de crear una promesa es:

Para crear una promesa deberemos poner
variable= new Promise(nombre-función-1)

Creamos una promesa e indicamos el nombre de una función, que contiene el código que se va a ejecutar cuando se realiza la promesa (la acción que prometemos que vamos a desarrollar).

ejemplo, creamos una promesa, que cuando se realice se va a ejecutar la función tratar

| | |
|-----|------------------------------------|
| 001 | let mipromesa=new Promise(tratar); |
|-----|------------------------------------|

A continuación, a la variable de la promesa le vamos a aplicar los métodos then y catch.

variable

```
.then(función-resuelve)  
.catch(función-error);
```

En el método **then** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones prometidas.

En el método **catch** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones que se van a realizar cuando se rechaza la promesa o hay un error en la misma.

| | |
|-----|---------------------------|
| 001 | mipromesa |
| 002 | .then (correcto) |
| 003 | .catch (errores) ; |

Vamos a declarar la función que se va a ejecutar cuando se realiza la promesa, lo que prometemos que vamos a hacer.

```
function nombre-función-1(función-resolver, función-rechazar){  
    cuerpo de la función.  
}
```

Esta función va a recibir dos parámetros:

- El primer parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se acepta la promesa, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de resolución de la promesa.
- El segundo parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se rechaza la promesa o cuando existe un error, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de rechazo o error de la promesa.

En el cuerpo de la función vamos a indicar que deseamos hacer y en un momento llamaremos a la función que resuelve la promesa, en este caso se la puede pasar varios parámetros, y en otro momento llamaremos a la función que rechaza la promesa.

Ejemplo: en este caso cogemos el valor de dos cajas de texto, si el valor de ambas es distinto de la cadena vacía llamamos a la función resolver con el valor de las dos cajas de texto y si no se llama a la función rechazar.

| | |
|-----|---|
| 001 | function tratar(resolver, rechazar){ |
| 002 | let nom=document.getElementById("nombre").value.trim(); |
| 003 | let ape=document.getElementById("apellidos").value.trim(); |
| 004 | if (nom != "" && ape != "") |
| 005 | resolver(nom , ape) |
| 006 | else |
| 007 | rechazar(); |
| 008 | } |

A continuación nos declaramos la función que resuelve la promesa

```
function función-resuelve([parametros]){  
    cuerpo-función  
}
```

Ejemplo: mostramos en una caja de texto la concatenación de los dos parámetros

| | |
|-----|--|
| 001 | function correcto(unos,dos) { |
| 002 | document.getElementById("resultado").value=unos + " " + dos; |
| 003 | } |

Y por último nos declaramos la función de rechazo de la promesa o de error de la promesa

```
function función-errores([parámetros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto un mensaje de error.

| | |
|-----|---|
| 001 | function errores() { |
| 002 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 003 | } |

Ahora vamos a ver el programa completo

promise111.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio() { |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", crear) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", crear); |
| 011 | } |
| 012 | function crear() { |
| 013 | let mipromesa= new Promise(tratar); |
| 014 | mipromesa |
| 015 | .then(correcto) |
| 016 | .catch(errores); |
| 017 | } |
| 018 | function tratar(resolver, rechazar){ |
| 019 | let nom=document.getElementById("nombre").value.trim(); |
| 020 | let ape=document.getElementById("apellidos").value.trim(); |
| 021 | if (nom!="" && ape!="") |
| 022 | resolver(nom + " " + ape) |
| 023 | else |
| 024 | rechazar(); |
| 025 | } |
| 026 | function correcto(unos,dos) { |
| 027 | document.getElementById("resultado").value=unos; |
| 028 | } |
| 029 | function errores() { |
| 030 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 031 | } |

Vamos a ver varios ejemplos del mismo programa haciéndolo de otra forma

promise112.js

| | |
|-----|--|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio() { |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", crear) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", crear); |
| 011 | } |
| 012 | function crear() { |
| 013 | let mipromesa= new Promise(|
| 014 | function (resolver, rechazar) { |
| 015 | let nom=document.getElementById("nombre").value.trim(); |
| 016 | let ape=document.getElementById("apellidos").value.trim(); |
| 017 | if (nom!="" && ape!="") |
| 018 | resolver(nom + " " + ape) |
| 019 | else |

| | |
|-----|--|
| 020 | rechazar(); |
| 021 | } |
| 022 |); |
| 023 | mipromesa |
| 024 | .then(function (uno) { |
| 025 | document.getElementById("resultado").value=uno; |
| 026 | }) |
| 027 | .catch(function () { |
| 028 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 029 | }); |
| 030 | } |

promise113.js

| | |
|-----|--|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio() { |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", crear) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", crear); |
| 011 | } |
| 012 | function crear() { |
| 013 | let mipromesa= new Promise((resolver, rechazar)=>{ |
| 014 | let nom=document.getElementById("nombre").value.trim(); |
| 015 | let ape=document.getElementById("apellidos").value.trim(); |
| 016 | if (nom!="" && ape!="") |
| 017 | resolver(nom + " " + ape) |
| 018 | else |
| 019 | rechazar(); |
| 020 | } |
| 021 |); |
| 022 | mipromesa |
| 023 | .then((uno) => { |
| 024 | document.getElementById("resultado").value=uno; |
| 025 | }) |
| 026 | .catch(()=>{ |
| 027 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 028 | }); |
| 029 | } |

promise114.js

| | |
|-----|--|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load", inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload", inicio); |
| 005 | function inicio() { |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click", crear) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick", crear); |
| 011 | } |
| 012 | function crear() { |
| 013 | let mipromesa= new Promise((resolver, rechazar)=>{ |
| 014 | let nom=document.getElementById("nombre").value.trim(); |
| 015 | let ape=document.getElementById("apellidos").value.trim(); |
| 016 | if (nom!="" && ape!="") |
| 017 | resolver(nom + " " + ape) |
| 018 | else |
| 019 | rechazar(); |
| 020 | } |
| 021 |); |
| 022 | mipromesa |
| 023 | .then(uno => { |
| 024 | document.getElementById("resultado").value=uno; |
| 025 | }) |
| 026 | .catch(()=>{ |

| | |
|-----|---|
| 027 | <code>document.getElementById("resultado").value="Al menos uno de los campos</code> |
| 028 | <code>está vacío";</code> |
| 028 | <code>});</code> |
| 029 | <code>}</code> |

Una tercera forma de crear una promesa es:

En una función devolvemos una promesa que creamos en ese preciso momento.

```
function nombre-función(){
    return new Promise(nombre-función-1);
}
```

Creamos una promesa e indicamos el nombre de una función, que contiene el código que se va a ejecutar cuando se realiza la promesa (la acción que prometemos que vamos a desarrollar) y devolvemos la promesa creada en la misma línea.

| | |
|-----|--|
| 001 | <code>function crear(){</code> |
| 002 | <code> return new Promise(tratar);</code> |
| 003 | <code>}</code> |

Deberemos crearnos una función que va a ser la encargada de prometer la promesa, esto es, indicar que se realice la promesa.

```
function nombre-función(){
    crearPromesa().then(función-resuelve).catch(función-error);
}
```

O bien

```
function nombre-función(){
    crearPromesa()
        .then(función-resuelve)
        .catch(función-error);
}
```

En el método **then** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones prometidas.

En el método **catch** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones que se van a realizar cuando se rechaza la promesa o hay un error en la misma.

| | |
|-----|--------------------------------------|
| 001 | <code>function enviar(){</code> |
| 002 | <code> crear()</code> |
| 003 | <code> .then(correcto)</code> |
| 004 | <code> .catcherrores)</code> |
| 005 | <code>}</code> |

Vamos a declarar la función que se va a ejecutar cuando se realiza la promesa, lo que prometemos que vamos a hacer.

```
function nombre-función-1(función-resolver, función-rechazar){
    cuerpo de la función.
}
```

Esta función va a recibir dos parámetros:

- El primer parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se acepta la promesa, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de resolución de la promesa.
- El segundo parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se rechaza la promesa o cuando existe un error, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de rechazo o error de la promesa.

En el cuerpo de la función vamos a indicar que deseamos hacer y en un momento llamaremos a la función que resuelve la promesa, en este caso se la puede pasar varios parámetros, y en otro momento llamaremos a la función que rechaza la promesa.

Ejemplo: en este caso cogemos el valor de dos cajas de texto, si el valor de ambas es distinto de la cadena vacía llamamos a la función resolver con el valor de las dos cajas de texto y si no se llama a la función rechazar.

| | |
|-----|---|
| 001 | <code>function tratar(resolver, rechazar){</code> |
| 002 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 003 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 004 | <code>if(nom != "" && ape != "")</code> |
| 005 | <code>resolver(nom , ape)</code> |
| 006 | <code>else</code> |
| 007 | <code>rechazar();</code> |
| 008 | <code>}</code> |

A continuación nos declaramos la función que resuelve la promesa

```
function función-resuelve([parametros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto la concatenación de los dos parámetros

| | |
|-----|--|
| 001 | <code>function correcto(uno,dos){</code> |
| 002 | <code>document.getElementById("resultado").value=uno + " " + dos;</code> |
| 003 | <code>}</code> |

Y por último nos declaramos la función de rechazo de la promesa o de error de la promesa

```
function función-error([parámetros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto un mensaje de error.

| | |
|-----|--|
| 001 | <code>function errores(){</code> |
| 002 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 003 | <code>}</code> |

Ahora vamos a ver el programa completo

promise121.js

| | |
|-----|--|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code>window.addEventListener("load",inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code>window.attachEvent("onload",inicio);</code> |
| 005 | <code>function inicio(){</code> |
| 006 | <code>let boton=document.getElementById("poner");</code> |
| 007 | <code>if (document.addEventListener)</code> |
| 008 | <code>boton.addEventListener("click",enviar)</code> |
| 009 | <code>else if (document.attachEvent)</code> |
| 010 | <code>boton.attachEvent("onclick",enviar);</code> |

| | |
|-----|---|
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then(correcto) |
| 015 | .catcherrores) |
| 016 | } |
| 017 | function crear(){ |
| 018 | return new Promise(tratar); |
| 019 | } |
| 020 | function tratar(resolver, rechazar){ |
| 021 | let nom=document.getElementById("nombre").value.trim(); |
| 022 | let ape=document.getElementById("apellidos").value.trim(); |
| 023 | if (nom!="" && ape!="") |
| 024 | resolver(nom + " " + ape) |
| 025 | else |
| 026 | rechazar(); |
| 027 | } |
| 028 | function correcto(unos){ |
| 029 | document.getElementById("resultado").value=unos; |
| 030 | } |
| 031 | function errores(){ |
| 032 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 033 | } |

promise122.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load",inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload",inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click",enviar) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick",enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then(correcto) |
| 015 | .catcherrores) |
| 016 | } |
| 017 | function crear(){ |
| 018 | return new Promise(|
| 019 | function (resolver, rechazar){ |
| 020 | let nom=document.getElementById("nombre").value.trim(); |
| 021 | let ape=document.getElementById("apellidos").value.trim(); |
| 022 | if (nom!="" && ape!="") |
| 023 | resolver(nom+ " " +ape) // solo se puede enviar un parámetro |
| 024 | else |
| 025 | rechazar(); |
| 026 | } |
| 027 |); |
| 028 | } |
| 029 | function correcto(unos){ |
| 030 | document.getElementById("resultado").value=unos; |
| 031 | } |
| 032 | function errores(){ |
| 033 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 034 | } |

promise123.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load",inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload",inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click",enviar) |
| 009 | else if (document.attachEvent) |

| | |
|-----|--|
| 010 | boton.attachEvent("onclick",enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then() |
| 015 | function (uno){ |
| 016 | document.getElementById("resultado").value=uno; |
| 017 | } |
| 018 | .catch() |
| 019 | function (){ |
| 020 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 021 | }); |
| 022 | } |
| 023 | function crear(){ |
| 024 | return new Promise(|
| 025 | function (resolver, rechazar){ |
| 026 | let nom=document.getElementById("nombre").value.trim(); |
| 027 | let ape=document.getElementById("apellidos").value.trim(); |
| 028 | if (nom!="" && ape!="") |
| 029 | resolver(nom + " " + ape) |
| 030 | else |
| 031 | rechazar(); |
| 032 | }); |
| 033 | } |

promise124.js

| | |
|-----|--|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load",inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload",inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click",enviar) |
| 009 | else if (document.attachEvent) |
| 010 | boton.attachEvent("onclick",enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then() |
| 015 | (uno)=> |
| 016 | document.getElementById("resultado").value=uno; |
| 017 | } |
| 018 | .catch() |
| 019 | ()=> |
| 020 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 021 | }); |
| 022 | } |
| 023 | function crear(){ |
| 024 | return new Promise(|
| 025 | (resolver, rechazar)=> |
| 026 | let nom=document.getElementById("nombre").value.trim(); |
| 027 | let ape=document.getElementById("apellidos").value.trim(); |
| 028 | if (nom!="" && ape!="") |
| 029 | resolver(nom + " " + ape) |
| 030 | else |
| 031 | rechazar(); |
| 032 | }); |
| 033 | } |

promise125.js

| | |
|-----|---|
| 001 | if (document.addEventListener) |
| 002 | window.addEventListener("load",inicio) |
| 003 | else if (document.attachEvent) |
| 004 | window.attachEvent("onload",inicio); |
| 005 | function inicio(){ |
| 006 | let boton=document.getElementById("poner"); |
| 007 | if (document.addEventListener) |
| 008 | boton.addEventListener("click",enviar) |
| 009 | else if (document.attachEvent) |



| | |
|-----|---|
| 010 | boton.attachEvent("onclick",enviar); |
| 011 | } |
| 012 | function enviar(){ |
| 013 | crear() |
| 014 | .then({ |
| 015 | uno =>{ |
| 016 | document.getElementById("resultado").value=uno; |
| 017 | }) |
| 018 | .catch({ |
| 019 | ()=>{ |
| 020 | document.getElementById("resultado").value="Al menos uno de los campos está vacío"; |
| 021 | }); |
| 022 | } |
| 023 | function crear(){ |
| 024 | return new Promise({ |
| 025 | (resolver, rechazar)=>{ |
| 026 | let nom=document.getElementById("nombre").value.trim(); |
| 027 | let ape=document.getElementById("apellidos").value.trim(); |
| 028 | if (nom!="" && ape!="") |
| 029 | resolver(nom + " " + ape) |
| 030 | else |
| 031 | rechazar(); |
| 032 | }); |
| 033 | } |

Una cuarta forma de crear una promesa es:

Para crear una promesa deberemos poner
variable= new Promise(nombre-función-1)
.then(función-resuelve)
.catch(función-error);

Creamos una promesa e indicamos el nombre de una función, que contiene el código que se va a ejecutar cuando se realiza la promesa (la acción que prometemos que vamos a desarrollar), con lo métodos **then** y **catch**.

En el método **then** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones prometidas.

En el método **catch** deberemos poner el nombre de una función, que está declarada más adelante y que se va a encargar de realizar las acciones que se van a realizar cuando se rechaza la promesa o hay un error en la misma.

| | |
|-----|------------------------------------|
| 001 | function enviar(){ |
| 002 | let mipromesa= new Promise(tratar) |
| 003 | .then(correcto) |
| 004 | .catch(errones) |
| 005 | } |

Vamos a declarar la función que se va a ejecutar cuando se realiza la promesa, lo que prometemos que vamos a hacer.

function nombre-función-1(función-resolver, función-rechazar){
cuerpo de la función.
}

Esta función va a recibir dos parámetros:

- El primer parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se acepta la promesa, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de resolución de la promesa.

- El segundo parámetro se corresponde con el nombre simbólico de una función, que se va a mandar ejecutar cuando se rechaza la promesa o cuando existe un error, no existe una función con ese nombre, solamente se utiliza en esta función para hacer la llamada a la función de rechazo o error de la promesa.

En el cuerpo de la función vamos a indicar que deseamos hacer y en un momento llamaremos a la función que resuelve la promesa, en este caso se la puede pasar varios parámetros, y en otro momento llamaremos a la función que rechaza la promesa.

Ejemplo: en este caso cogemos el valor de dos cajas de texto, si el valor de ambas es distinto de la cadena vacía llamamos a la función resolver con el valor de las dos cajas de texto y si no se llama a la función rechazar.

| | |
|-----|---|
| 001 | <code>function tratar(resolver, rechazar){</code> |
| 002 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 003 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 004 | <code>if(nom != "" && ape != "")</code> |
| 005 | <code>resolver(nom , ape)</code> |
| 006 | <code>else</code> |
| 007 | <code>rechazar();</code> |
| 008 | <code>}</code> |

A continuación nos declaramos la función que resuelve la promesa

```
function función-resuelve([parametros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto la concatenación de los dos parámetros

| | |
|-----|--|
| 001 | <code>function correcto(uno,dos) {</code> |
| 002 | <code>document.getElementById("resultado").value=uno + " " + dos;</code> |
| 003 | <code>}</code> |

Y por último nos declaramos la función de rechazo de la promesa o de error de la promesa

```
function función-error([parámetros]){
    cuerpo-función
}
```

Ejemplo: mostramos en una caja de texto un mensaje de error.

| | |
|-----|--|
| 001 | <code>function errores() {</code> |
| 002 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 003 | <code>}</code> |

Ahora vamos a ver el programa completo

promise131.js

| | |
|-----|--|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code>window.addEventListener("load",inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code>window.attachEvent("onload",inicio);</code> |
| 005 | <code>function inicio(){</code> |
| 006 | <code>let boton=document.getElementById("poner");</code> |
| 007 | <code>if (document.addEventListener)</code> |
| 008 | <code>boton.addEventListener("click",enviar)</code> |
| 009 | <code>else if (document.attachEvent)</code> |
| 010 | <code>boton.attachEvent("onclick",enviar);</code> |
| 011 | <code>}</code> |
| 012 | <code>function enviar(){</code> |
| 013 | <code>let mipromesa= new Promise(tratar)</code> |
| 014 | <code>.then(correcto)</code> |
| 015 | <code>.catch(errores)</code> |
| 016 | <code>}</code> |



| | |
|-----|--|
| 017 | <code>function tratar(resolver, rechazar){</code> |
| 018 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 019 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 020 | <code>if (nom!="" && ape!="")</code> |
| 021 | <code>resolver(nom + " " + ape)</code> |
| 022 | <code>else</code> |
| 023 | <code>rechazar();</code> |
| 024 | <code>}</code> |
| 025 | <code>function correcto(unos,dos){</code> |
| 026 | <code>document.getElementById("resultado").value=unos;</code> |
| 027 | <code>}</code> |
| 028 | <code>function errores(){</code> |
| 029 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 030 | <code>}</code> |

promise132.js

| | |
|-----|--|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code>window.addEventListener("load",inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code>window.attachEvent("onload",inicio);</code> |
| 005 | <code>function inicio(){</code> |
| 006 | <code>let boton=document.getElementById("poner");</code> |
| 007 | <code>if (document.addEventListener)</code> |
| 008 | <code>boton.addEventListener("click",crear)</code> |
| 009 | <code>else if (document.attachEvent)</code> |
| 010 | <code>boton.attachEvent("onclick",crear);</code> |
| 011 | <code>}</code> |
| 012 | <code>function crear(){</code> |
| 013 | <code>let mipromesa= new Promise(</code> |
| 014 | <code>function (resolver, rechazar){</code> |
| 015 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 016 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 017 | <code>if (nom!="" && ape!="")</code> |
| 018 | <code>resolver(nom + " " + ape)</code> |
| 019 | <code>else</code> |
| 020 | <code>rechazar();</code> |
| 021 | <code>}</code> |
| 022 | <code>)</code> |
| 023 | <code>.then(function (uno){</code> |
| 024 | <code>document.getElementById("resultado").value=uno;</code> |
| 025 | <code>})</code> |
| 026 | <code>.catch(function (){</code> |
| 027 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 028 | <code>});</code> |
| 029 | <code>}</code> |

promise133.js

| | |
|-----|---|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code>window.addEventListener("load",inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code>window.attachEvent("onload",inicio);</code> |
| 005 | <code>function inicio(){</code> |
| 006 | <code>let boton=document.getElementById("poner");</code> |
| 007 | <code>if (document.addEventListener)</code> |
| 008 | <code>boton.addEventListener("click",crear)</code> |
| 009 | <code>else if (document.attachEvent)</code> |
| 010 | <code>boton.attachEvent("onclick",crear);</code> |
| 011 | <code>}</code> |
| 012 | <code>function crear(){</code> |
| 013 | <code>let mipromesa= new Promise(</code> |
| 014 | <code>(resolver, rechazar) => {</code> |
| 015 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 016 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 017 | <code>if (nom!="" && ape!="")</code> |
| 018 | <code>resolver(nom + " " + ape)</code> |
| 019 | <code>else</code> |
| 020 | <code>rechazar();</code> |
| 021 | <code>}</code> |
| 022 | <code>)</code> |
| 023 | <code>.then((uno)=>{</code> |

| | |
|-----|--|
| 024 | <code>document.getElementById("resultado").value=uno;</code> |
| 025 | <code>})</code> |
| 026 | <code>.catch(()=>{</code> |
| 027 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 028 | <code>});</code> |
| 029 | <code>}</code> |

promise134.js

| | |
|-----|--|
| 001 | <code>if (document.addEventListener)</code> |
| 002 | <code>window.addEventListener("load", inicio)</code> |
| 003 | <code>else if (document.attachEvent)</code> |
| 004 | <code>window.attachEvent("onload", inicio);</code> |
| 005 | <code>function inicio(){</code> |
| 006 | <code>let boton=document.getElementById("poner");</code> |
| 007 | <code>if (document.addEventListener)</code> |
| 008 | <code>boton.addEventListener("click", crear)</code> |
| 009 | <code>else if (document.attachEvent)</code> |
| 010 | <code>boton.attachEvent("onclick", crear);</code> |
| 011 | <code>}</code> |
| 012 | <code>function crear(){</code> |
| 013 | <code>let mipromesa= new Promise(</code> |
| 014 | <code>(resolver, rechazar) => {</code> |
| 015 | <code>let nom=document.getElementById("nombre").value.trim();</code> |
| 016 | <code>let ape=document.getElementById("apellidos").value.trim();</code> |
| 017 | <code>if (nom!="" && ape!="")</code> |
| 018 | <code>resolver(nom + " " + ape)</code> |
| 019 | <code>else</code> |
| 020 | <code>rechazar();</code> |
| 021 | <code>}</code> |
| 022 | <code>)</code> |
| 023 | <code>.then(unos =>{</code> |
| 024 | <code>document.getElementById("resultado").value=uno;</code> |
| 025 | <code>})</code> |
| 026 | <code>.catch(()=>{</code> |
| 027 | <code>document.getElementById("resultado").value="Al menos uno de los campos está vacío";</code> |
| 028 | <code>});</code> |
| 029 | <code>}</code> |