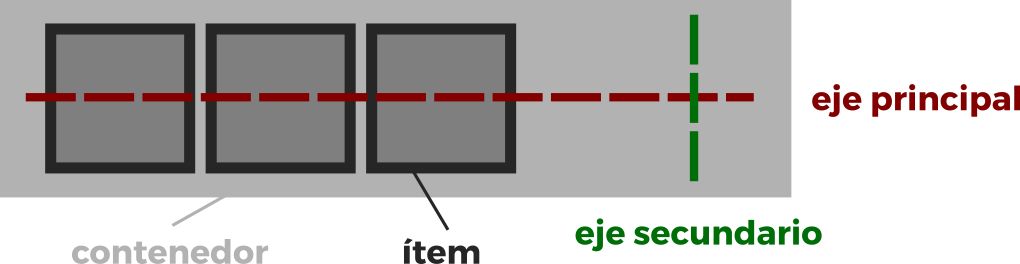
# Introducción

Tradicionalmente, en CSS se ha utilizado el posicionamiento (static, relative, absolute...), los elementos en línea o en bloque (y derivados) o los float, lo que a grandes rasgos no dejaba de ser un sistema de creación de diseños bastante tradicional que no encaja con los retos que tenemos hoy en día: sistemas de escritorio, dispositivos móviles, múltiples resoluciones, etc...

Flexbox es un sistema de elementos flexibles que llega con la idea de olvidar estos mecanismos y acostumbrarnos a una mecánica más potente, limpia y personalizable, en la que los elementos HTML se adaptan y colocan automáticamente y es más fácil personalizar los diseños. Está especialmente diseñado para crear, mediante CSS, estructuras de una sóla dimensión.

# Conceptos

Para empezar a utilizar flexbox lo primero que debemos hacer es conocer algunos de los elementos básicos de este nuevo esquema, que son los siguientes:



* Contenedor: Es el elemento padre que tendrá en su interior cada uno de los ítems flexibles. Observa que al contrario que muchas otras estructuras CSS, por norma general, en Flex establecemos las propiedades al elemento padre.
* Eje principal: Los contenedores flexibles tendrán una orientación principal específica. Por defecto, el eje principal del contenedor flexbox es en horizontal (en fila).
* Eje secundario: De la misma forma, los contenedores flexibles tendrán una orientación secundaria, perpendicular a la principal. Si la principal es en horizontal, la secundaria será en vertical (y viceversa).
* Ítem: Cada uno de los hijos que tendrá el contenedor en su interior.

Una vez tenemos claro esto, imaginemos el siguiente escenario:

  <div class="container"> <!-- Flex container -->

    <div class="item item-1">1</div> <!-- Flex items -->

    <div class="item item-2">2</div>

    <div class="item item-3">3</div>

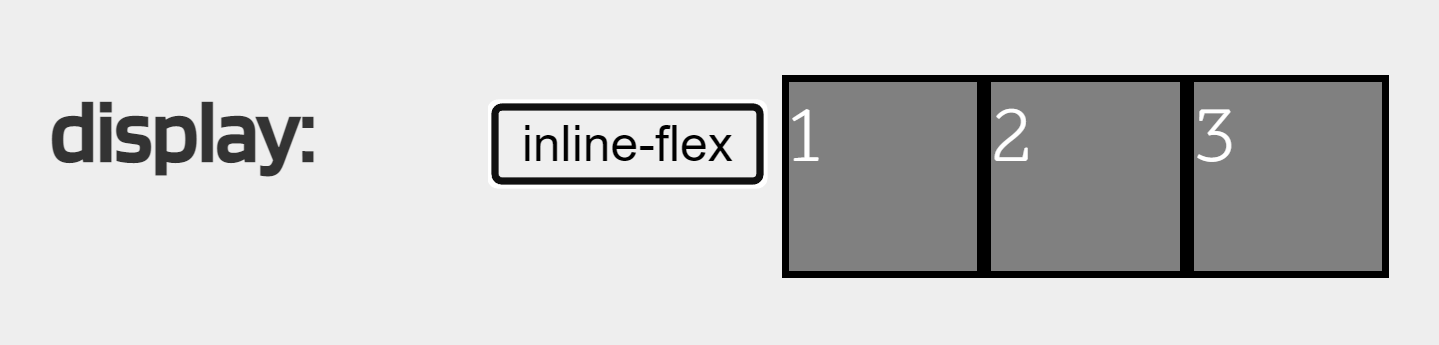
  </div>

Para activar el modo flexbox, hemos utilizado sobre el elemento contenedor la propiedad display  y especificamos el valor flex o inline-flex (dependiendo de cómo queramos que se comporte el contenedor):

|  |  |
| --- | --- |
| Tipo de elemento | Descripción |
| inline-flex | Establece un contenedor en línea, similar a **inline-block** (ocupa solo el contenido). |
| flex | Establece un contenedor en bloque, similar a **block** (ocupa todo el ancho del padre). |

Por defecto, y sólo con esto, observaremos que los elementos se disponen todos sobre una misma línea. Esto ocurre porque estamos utilizando el modo flexbox y estaremos trabajando con ítems flexibles básicos, garantizando que no se desbordarán ni mostrarán los problemas que, por ejemplo, tienen los porcentajes sobre elementos que no utilizan flexbox.





Dirección de los ejes

Existen dos propiedades principales para manipular la dirección y comportamiento de los ítems a lo largo del eje principal del contenedor. Son las siguientes:

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Significado |
| flex-direction | **row** | row-reverse | column | column-reverse | Cambia la orientación del eje principal. |

Mediante la propiedad flex-direction podemos modificar la dirección del eje principal del contenedor para que se oriente en horizontal (por defecto) o en vertical. Además, también podemos incluir el sufijo -reverse para indicar que coloque los ítems en orden inverso.

|  |  |
| --- | --- |
| Valor | Descripción |
| row | Establece la dirección del eje principal en horizontal. |
| row-reverse | Establece la dirección del eje principal en horizontal (invertido). |
| column | Establece la dirección del eje principal en vertical. |
| column-reverse | Establece la dirección del eje principal en vertical (invertido). |

Esto nos permite tener un control muy alto sobre el orden de los elementos en una página. Veamos la aplicación de estas propiedades sobre el ejemplo anterior, para modificar el flujo del eje principal del contenedor:

.container {

    display: flex;

    flex-direction: column;

    background: steelblue;

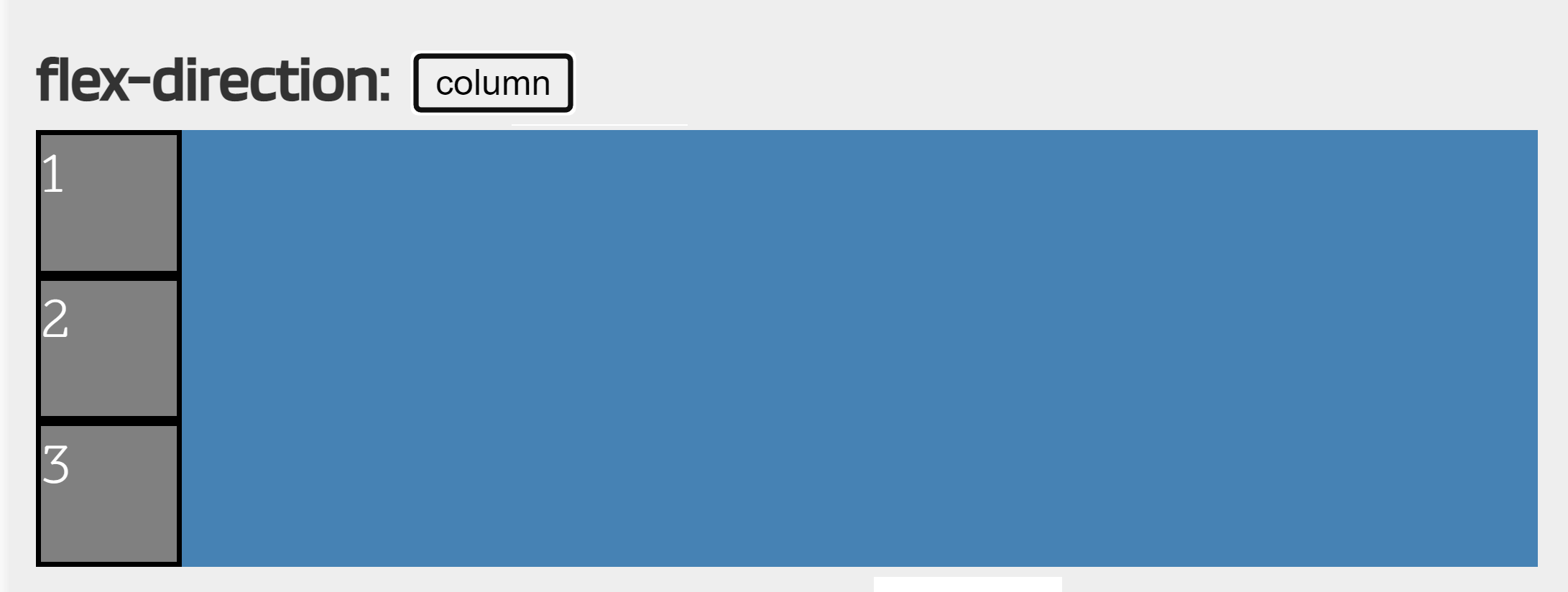
  }

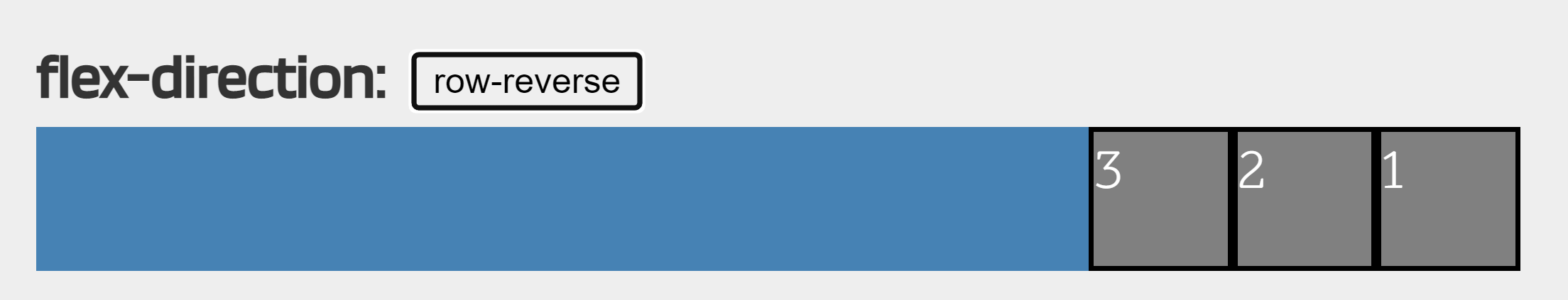
  .item {

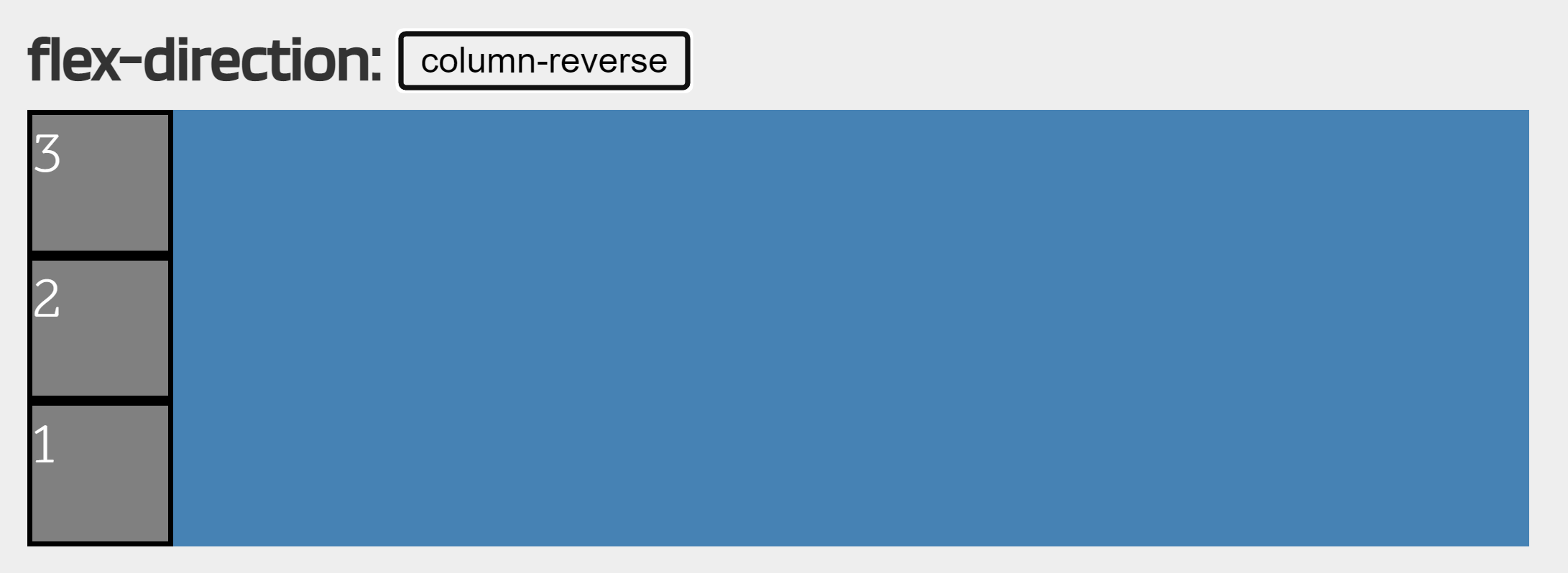
    background: grey;

A continuación, podemos ver ejemplos:









Contenedor flexbox multilínea

Por otro lado, existe otra propiedad llamada flex-wrap con la que podemos especificar el comportamiento del contenedor respecto a evitar que se desborde (nowrap, valor por defecto) o permitir que lo haga, en cuyo caso, estaríamos hablando de un contenedor flexbox multilínea.

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Significado |
| flex-wrap | **nowrap** | wrap | wrap-reverse | Evita o permite el desbordamiento (multilinea). |

Los valores que puede tomar esta propiedad, son las siguientes:

|  |  |
| --- | --- |
| Valor | Descripción |
| ****nowrap**** | Establece los ítems en una sola línea (no permite que se desborde el contenedor). |
| wrap | Establece los ítems en modo multilínea (permite que se desborde el contenedor). |
| wrap-reverse | Establece los ítems en modo multilínea, pero en dirección inversa. |

Teniendo en cuenta estos valores de la propiedad flex-wrap, podemos conseguir cosas como la siguiente:

.container {

    display: flex;

    flex-wrap: wrap;  /\* Comportamiento por defecto: nowrap \*/

    background: steelblue;

    width: 200px;

  }

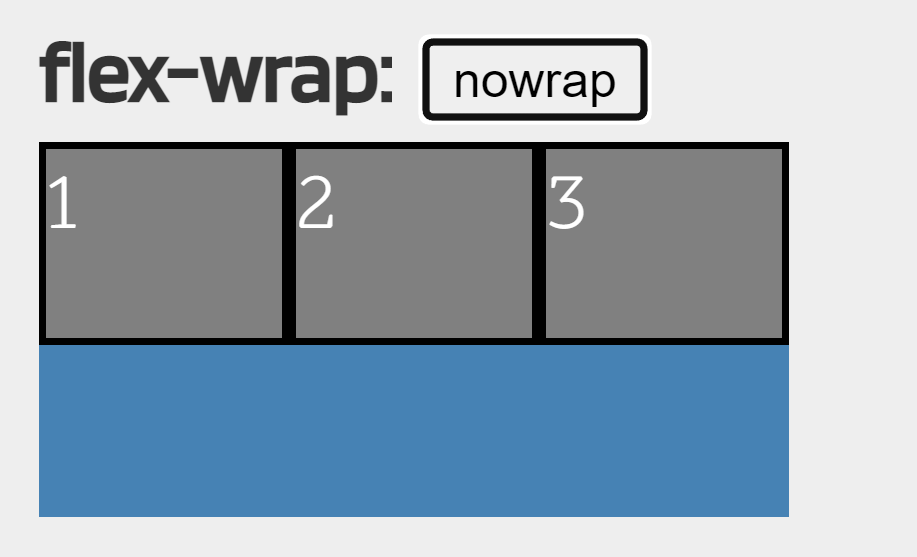
  .item {

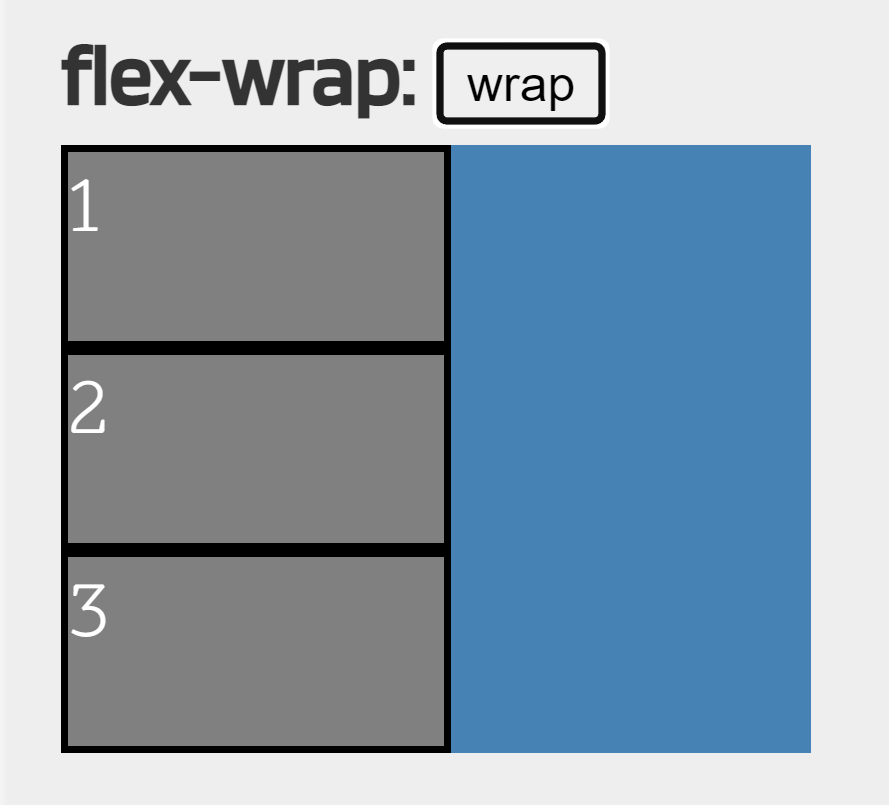
    background: grey;

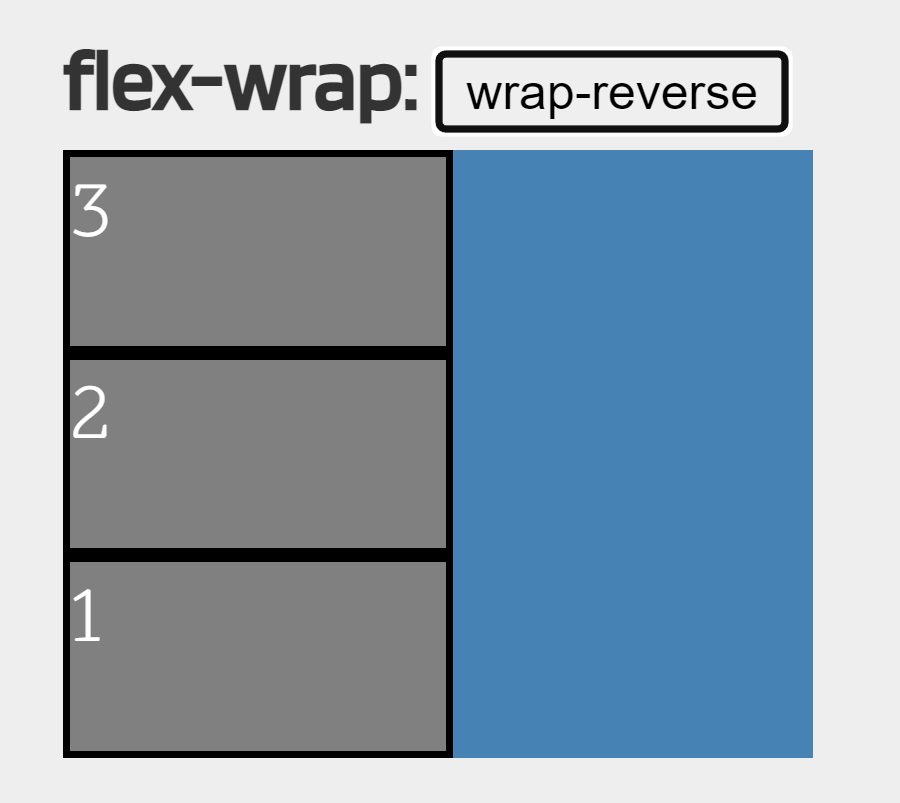
    width: 50%;

En el caso de especificar nowrap (u omitir la propiedad flex-wrap) en el contenedor, los 3 ítems se mostrarían en una misma línea del contenedor. En ese caso, cada ítem debería tener un 50% de ancho (o sea, 100px de los 200px del contenedor). Un tamaño de 100px por ítem, sumaría un total de 300px, que no cabrían en el contenedor de 200px, por lo que flexbox reajusta los ítems flexibles para que quepan todos en la misma línea, manteniendo las mismas proporciones.

Sin embargo, si especificamos wrap en la propiedad flex-wrap, lo que permitimos es que el contenedor se pueda desbordar, pasando a ser un contenedor multilínea, que mostraría el ítem 1 y 2 en la primera línea (con un tamaño de 100px cada uno) y el ítem 3 en la línea siguiente, dejando un espacio libre para un posible ítem 4.







Atajo: Dirección de los ejes

Recordemos que existe una propiedad de atajo (short-hand) llamada flex-flow, con la que podemos resumir los valores de las propiedades flex-direction y flex-wrap, especificándolas en una sola propiedad y ahorrándonos utilizar las propiedades concretas:

    .container {

        /\* flex-flow: <flex-direction> <flex-wrap>; \*/

        flex-flow: row wrap;

    }

Propiedades de alineación

Ahora que tenemos un control básico del contenedor de estos ítems flexibles, necesitamos conocer las propiedades existentes dentro de flexbox para disponer los ítems dependiendo de nuestro objetivo. Vamos a echar un vistazo a 4 propiedades interesantes para ello, la primera de ellas actúa en el eje principal, mientras que el resto en el eje secundario:

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Eje |
| justify-content | **flex-start** | flex-end | center | space-between | space-around | space-evenly | 1️⃣ |
| align-content | flex-start | flex-end | center | space-between | space-around | space-evenly | **stretch** | 2️⃣ |
| align-items | flex-start | flex-end | center | **stretch** | baseline | 2️⃣ |
| align-self | **auto** | flex-start | flex-end | center | stretch | baseline | 2️⃣ |

De esta pequeña lista, hay que centrarse en primer lugar en la primera y la tercera propiedad, que son las más importantes (las otras dos son casos particulares que explicaremos más adelante):

* justify-content: Se utiliza para alinear los ítems del eje principal (por defecto, el horizontal).
* align-items: Usada para alinear los ítems del eje secundario (por defecto, el vertical).

**Sobre el eje principal**

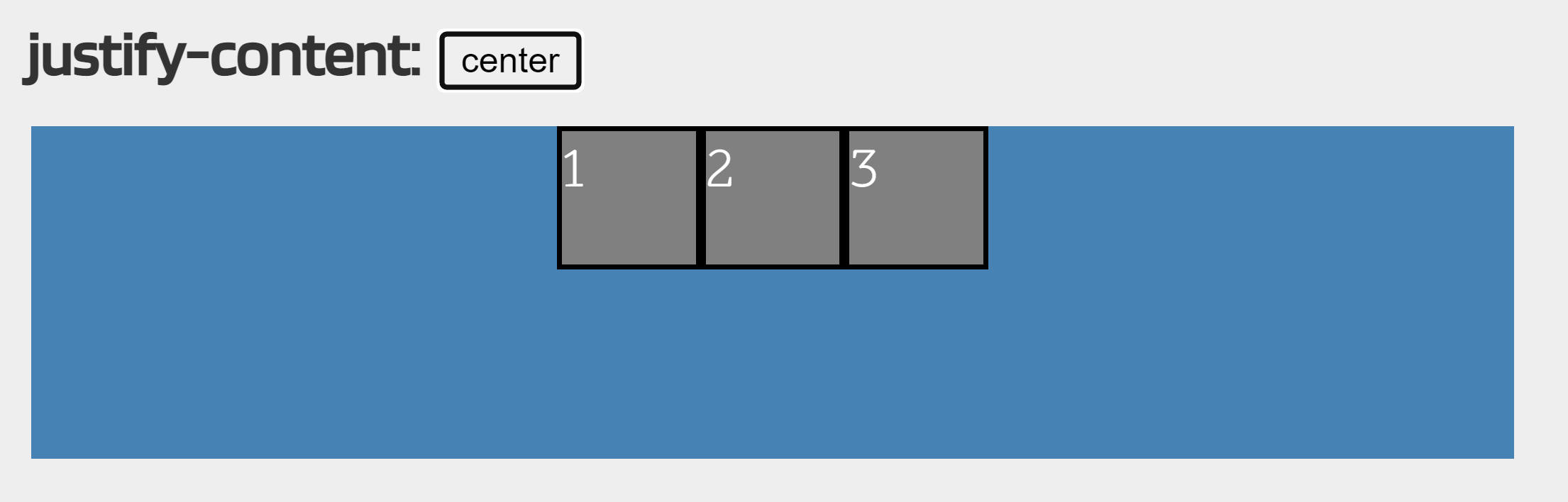
La primera propiedad, justify-content, sirve para colocar los ítems de un contenedor mediante una disposición concreta a lo largo del eje principal:

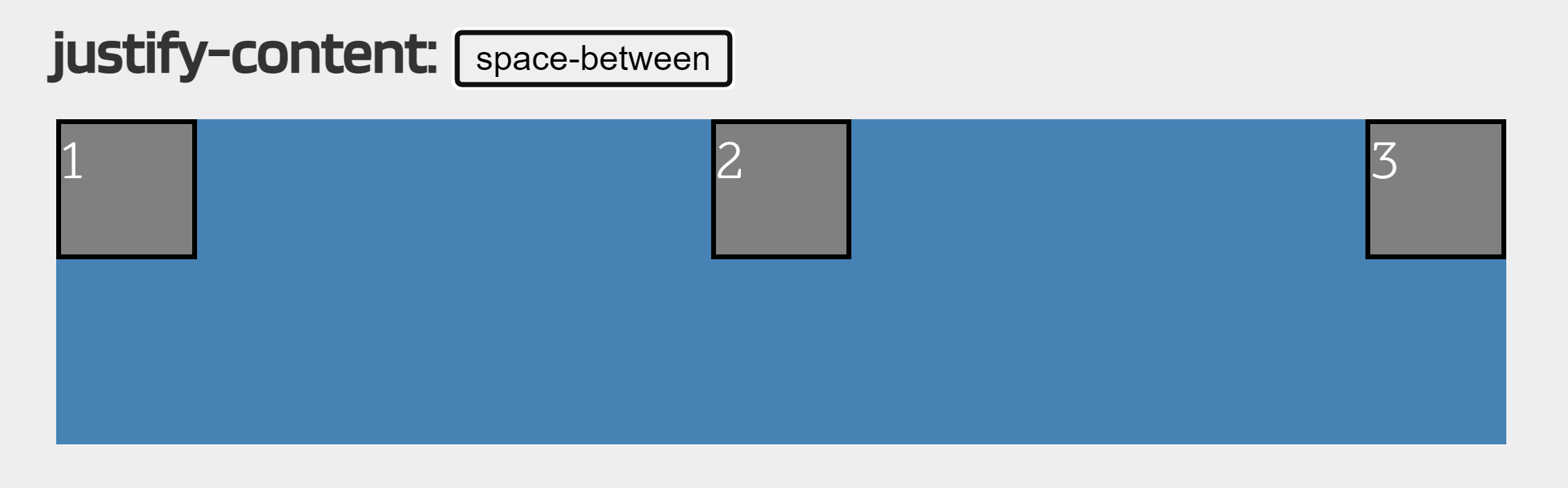
|  |  |
| --- | --- |
| Valor | Descripción |
| **flex-start** | Agrupa los ítems al principio del eje principal. |
| flex-end | Agrupa los ítems al final del eje principal. |
| center | Agrupa los ítems al centro del eje principal. |
| space-between | Distribuye los ítems dejando el máximo espacio para separarlos. |
| space-around | Distribuye los ítems dejando el mismo espacio alrededor de ellos (izq/dcha). |
| space-evenly | Distribuye los ítems dejando el mismo espacio (solapado) a izquierda y derecha. |

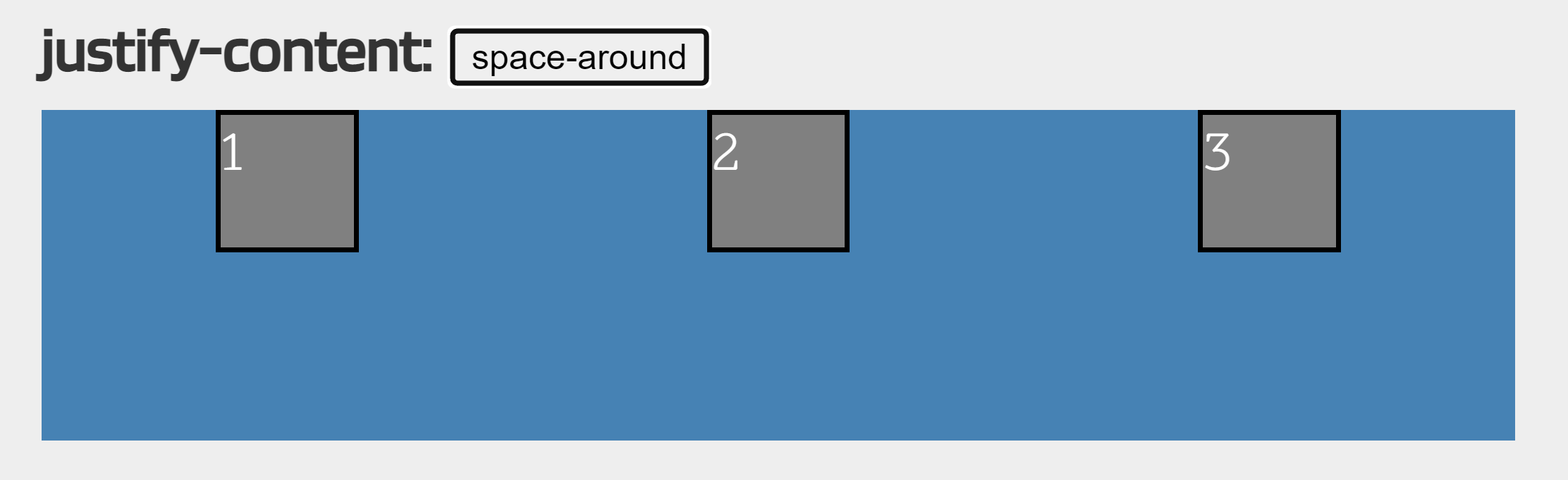
Con cada uno de estos valores, modificaremos la disposición de los ítems del contenedor donde se aplica, pasando a colocarse como se ve en el ejemplo interactivo siguiente (nótense los números para observar el orden de cada ítem):

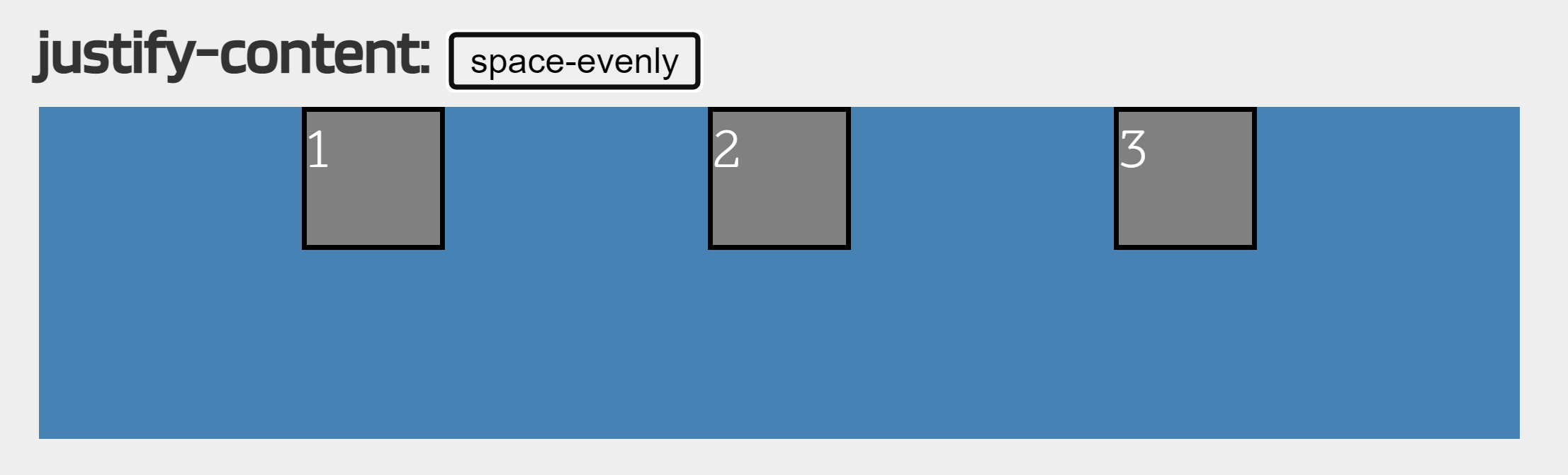












Una vez entendido este caso, debemos atender a la propiedad align-content, que es un caso particular del anterior. Nos servirá cuando estemos tratando con un contenedor flex multilínea, que es un contenedor en el que los ítems no caben en el ancho disponible, y por lo tanto, el eje principal se divide en múltiples líneas (por ejemplo, usando flex-wrap: wrap).

De esta forma, align-content servirá para alinear cada una de las líneas del contenedor multilínea. Los valores que puede tomar son los siguientes:

|  |  |
| --- | --- |
| Valor | Descripción |
| flex-start | Agrupa los ítems al principio del eje principal. |
| flex-end | Agrupa los ítems al final del eje principal. |
| center | Agrupa los ítems al centro del eje principal. |
| space-between | Distribuye los ítems desde el inicio hasta el final. |
| space-around | Distribuye los ítems dejando el mismo espacio a los lados de cada uno. |
| ****stretch**** | Estira los ítems para ocupar de forma equitativa todo el espacio. |

Con estos valores, vemos como cambiamos la disposición en vertical (porque partimos de un ejemplo en el que estamos utilizando flex-direction: row, y el eje principal es horizontal) de los ítems que están dentro de un contenedor multilínea.

En el ejemplo siguiente, veremos que al indicar un contenedor de 200 píxels de alto con ítems de 50px de alto y un flex-wrap establecido para tener contenedores multilínea, podemos utilizar la propiedad align-content para alinear los ítems de forma vertical de modo que se queden en la zona inferior del contenedor:

    .container {

        background: #CCC;

        display: flex;

        width: 200px;

        height: 200px;

        flex-wrap: wrap;

        align-content: flex-end;

    }

    .item {

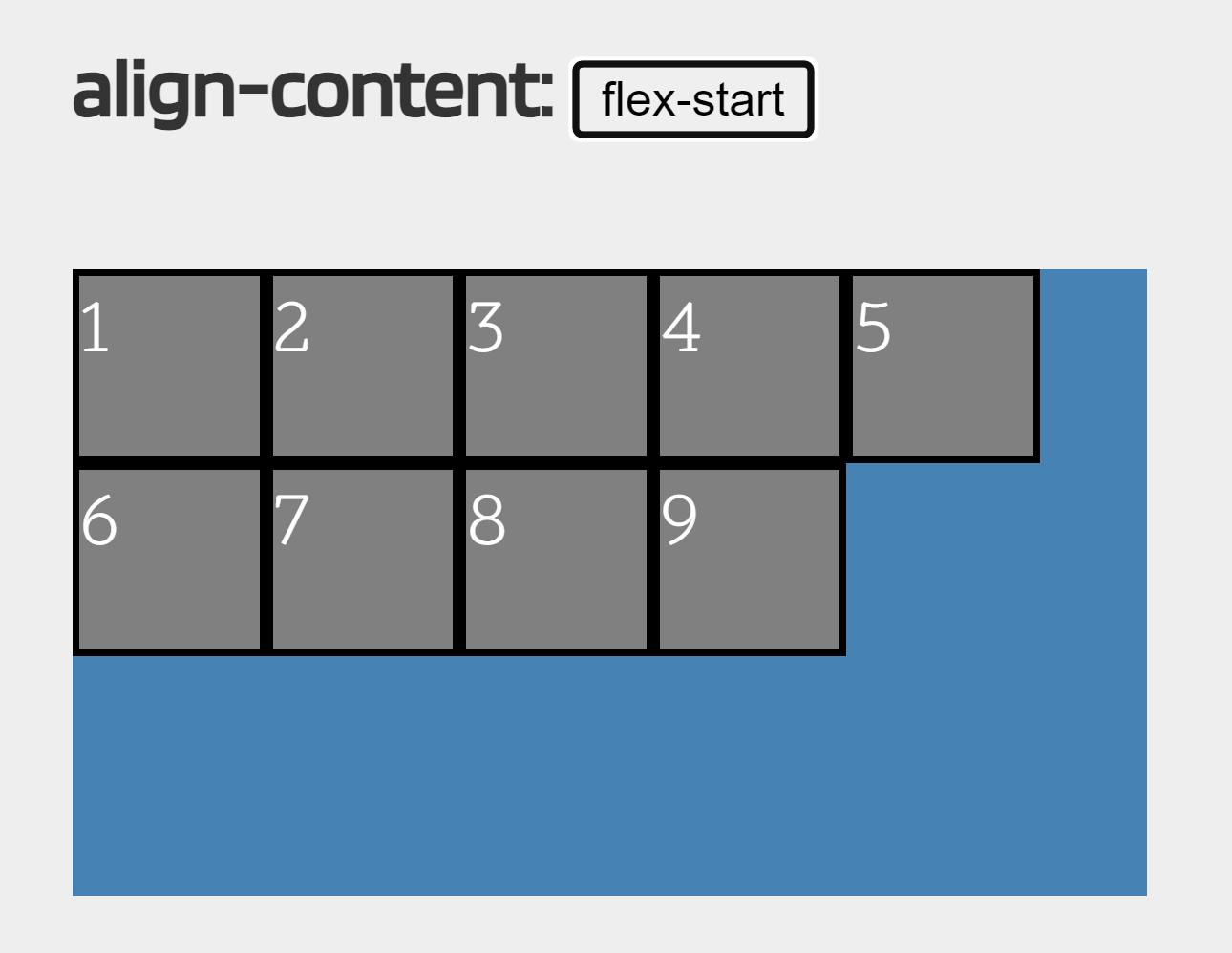
        background: #777;

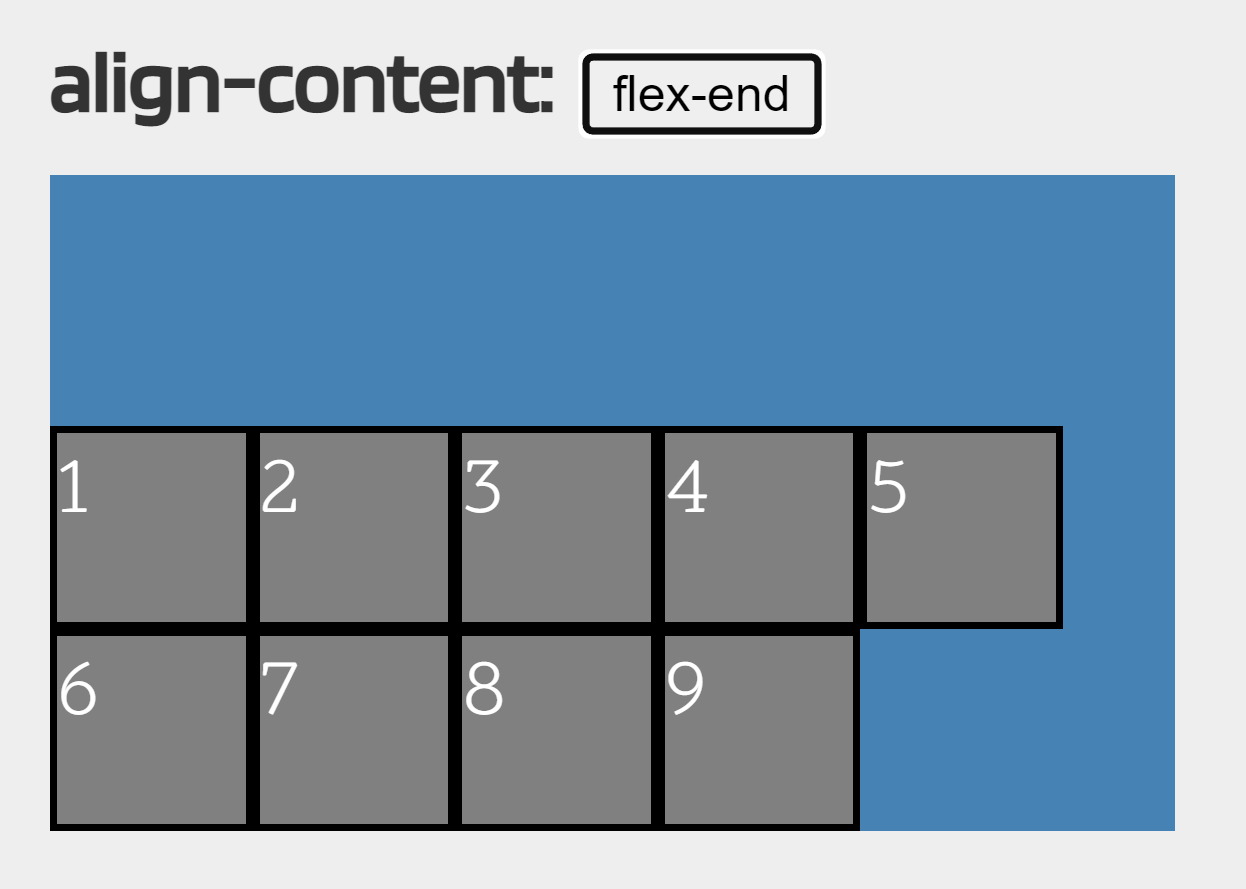
        width: 50%;

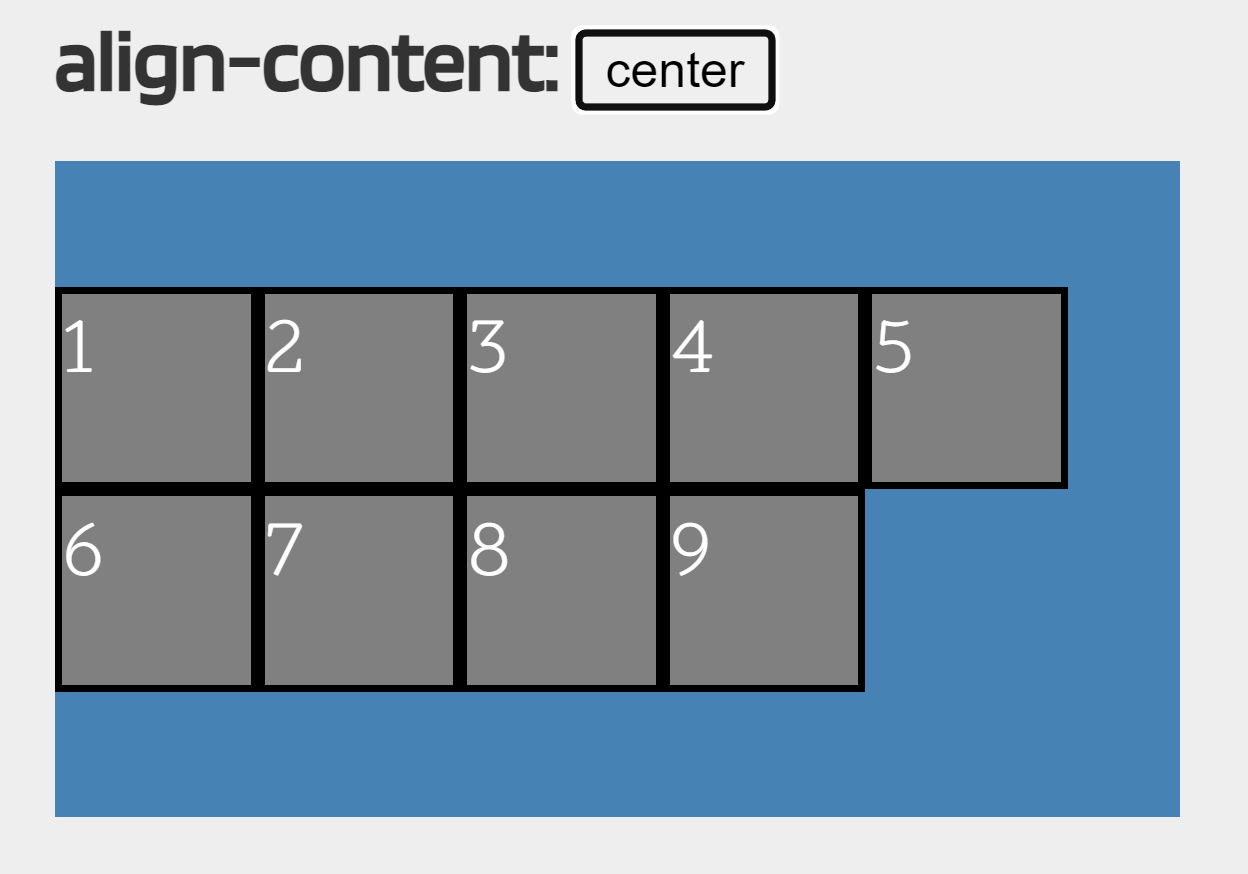
        height: 50px;

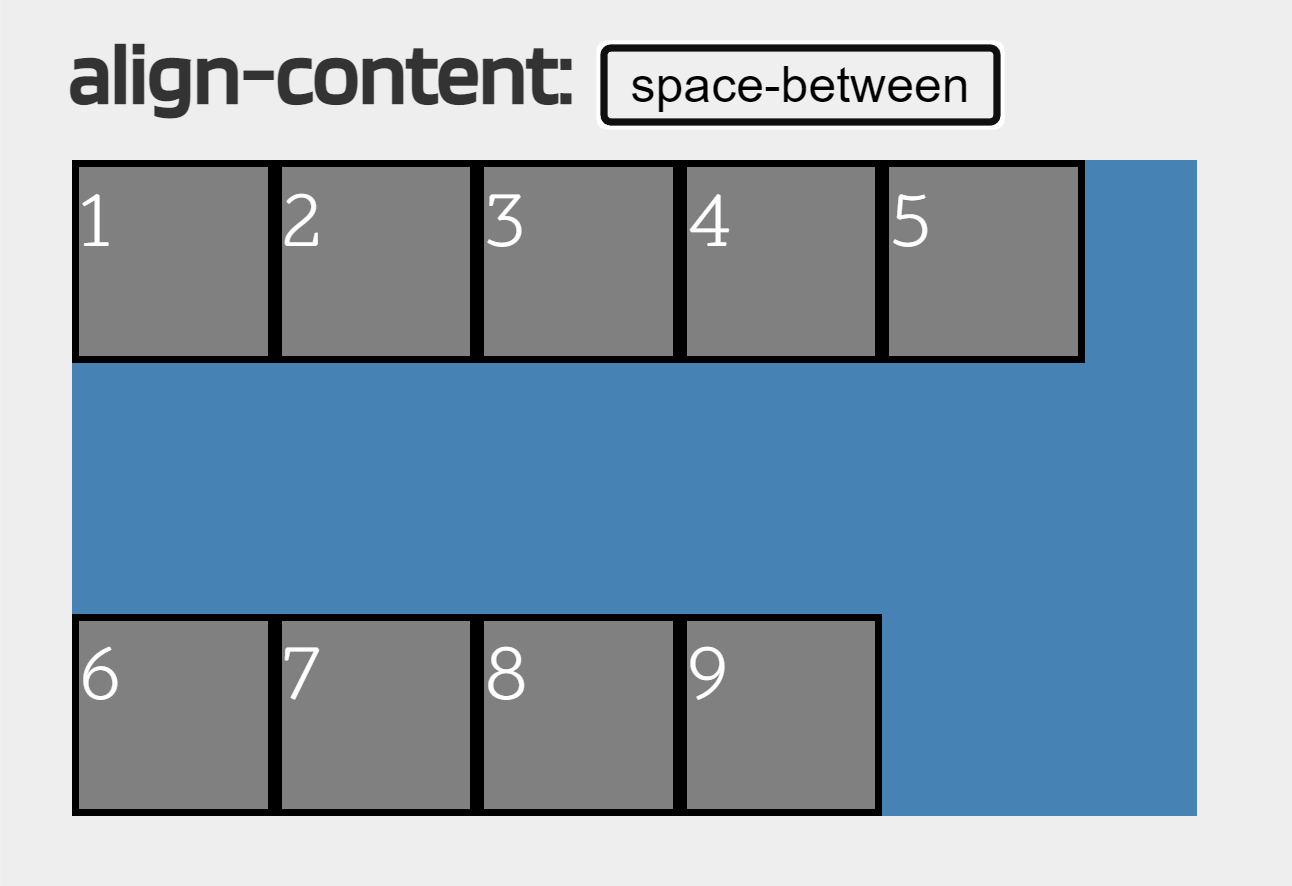
    }

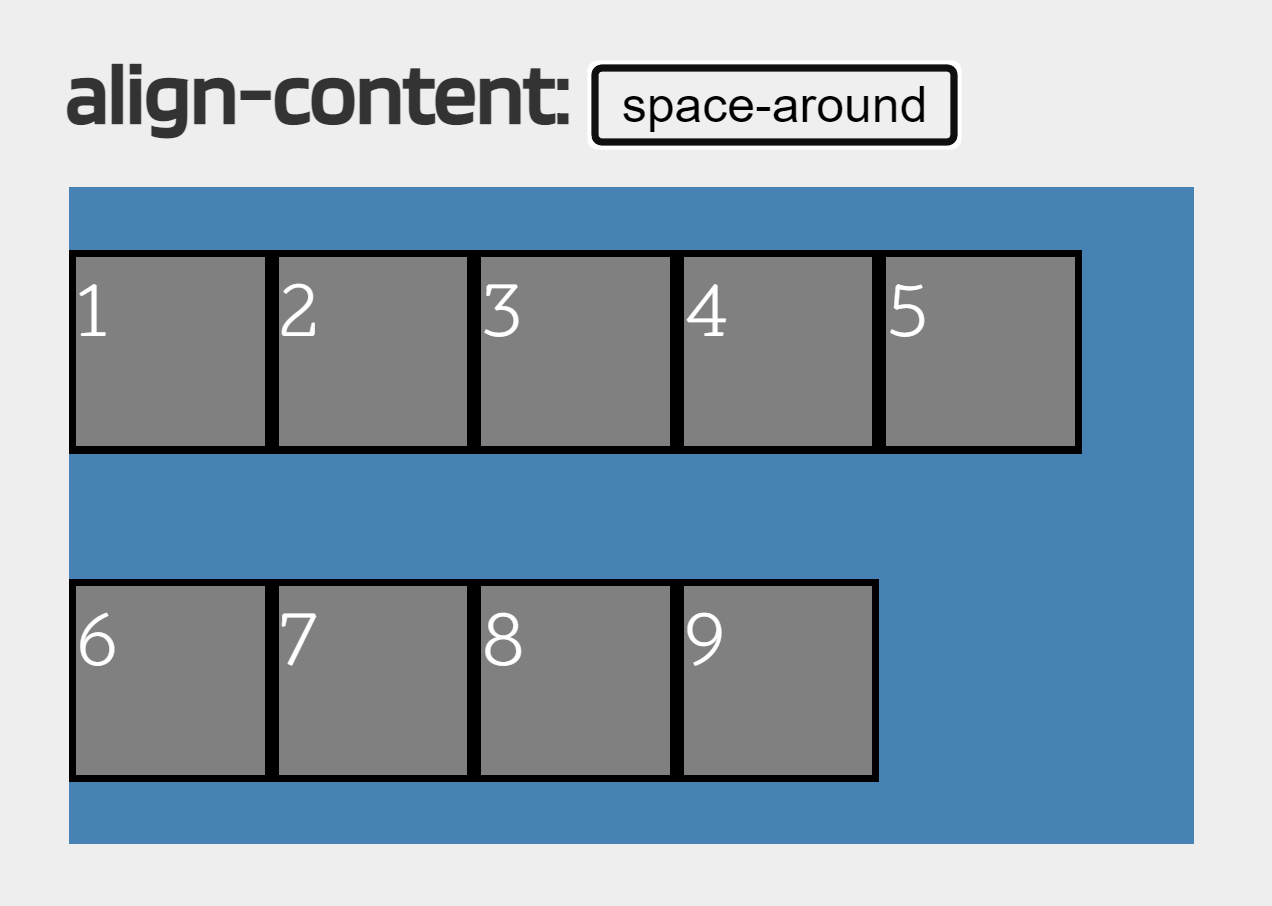
Observa cómo funciona la propiedad align-content en el siguiente ejemplo interactivo:

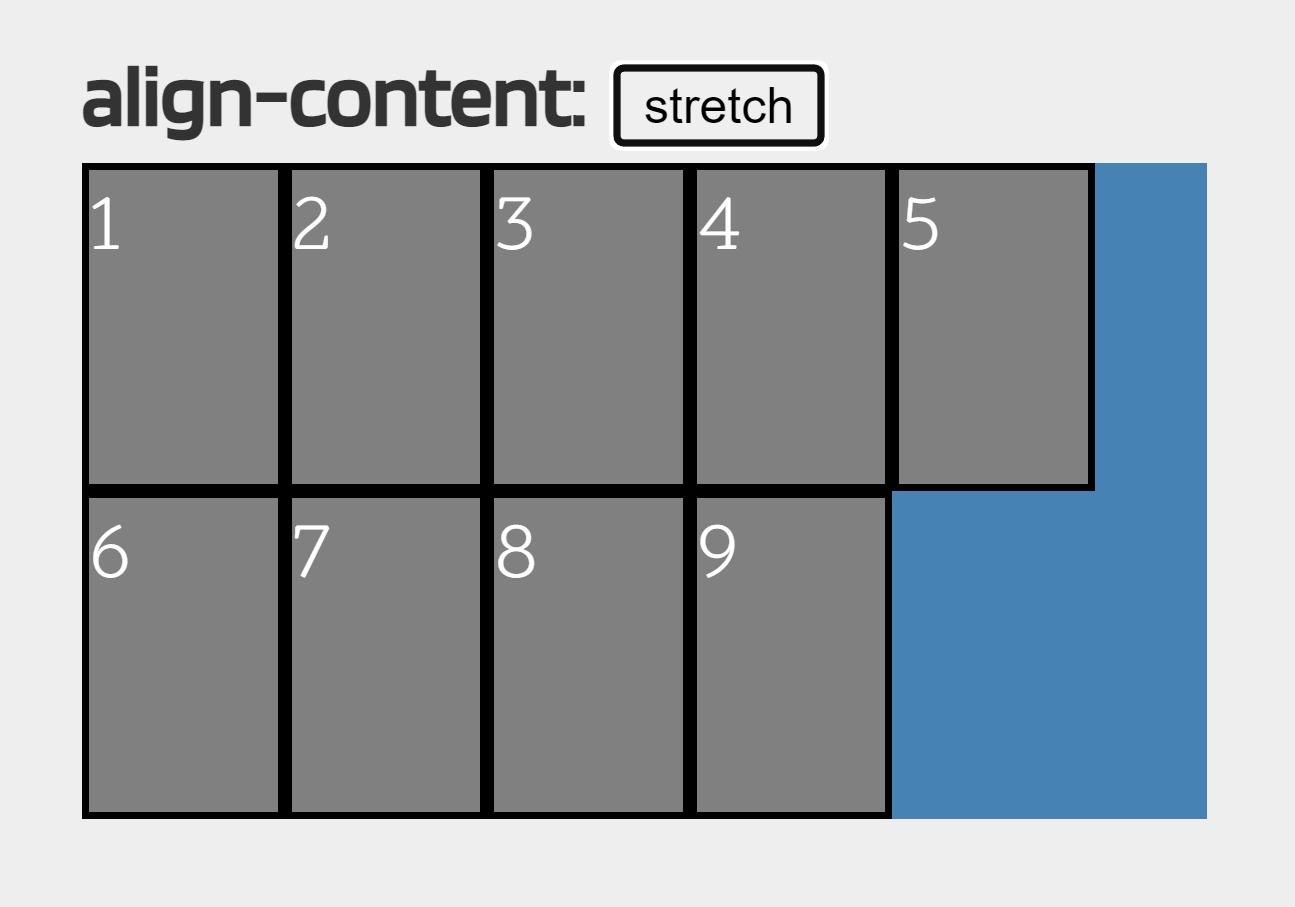


****

****

****

****

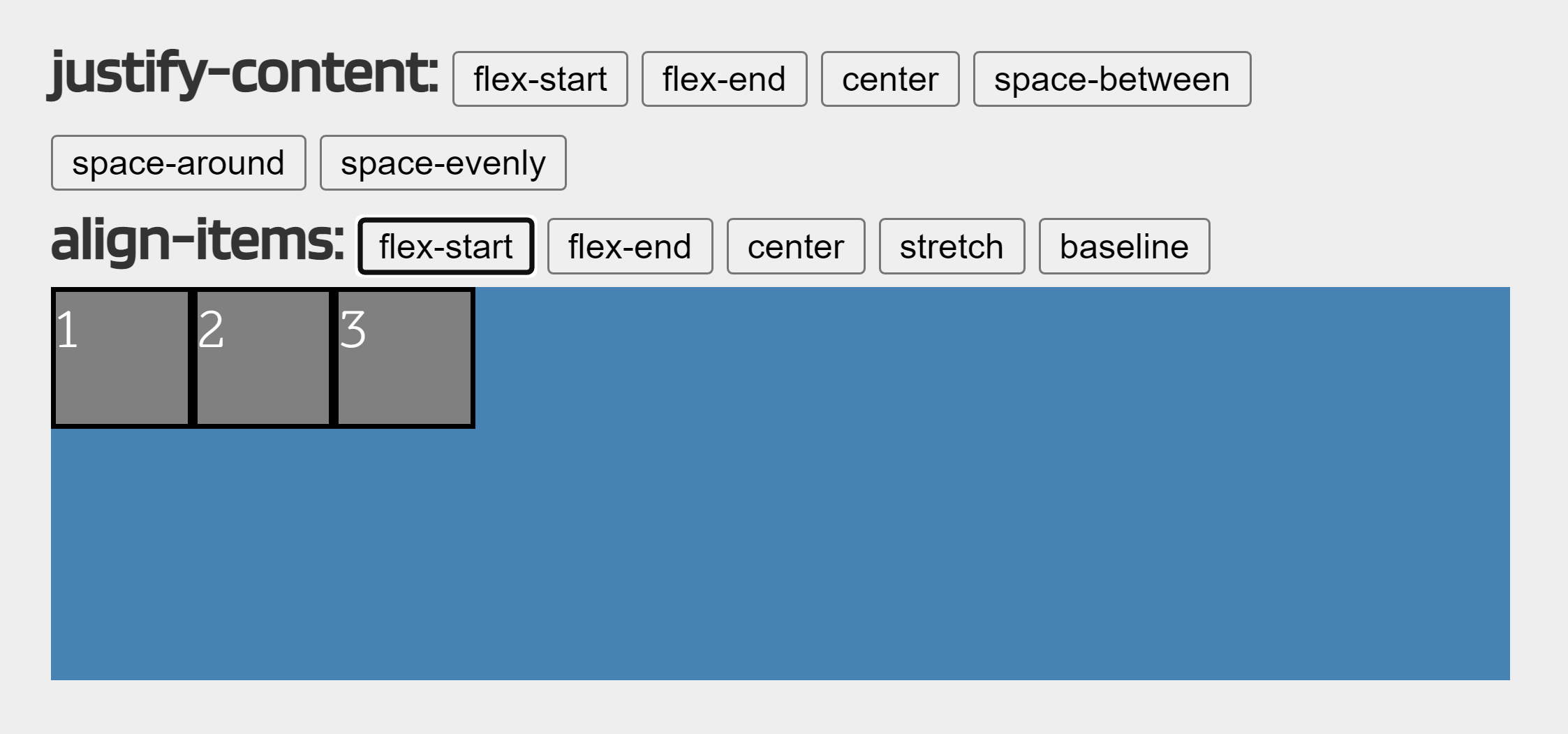
****

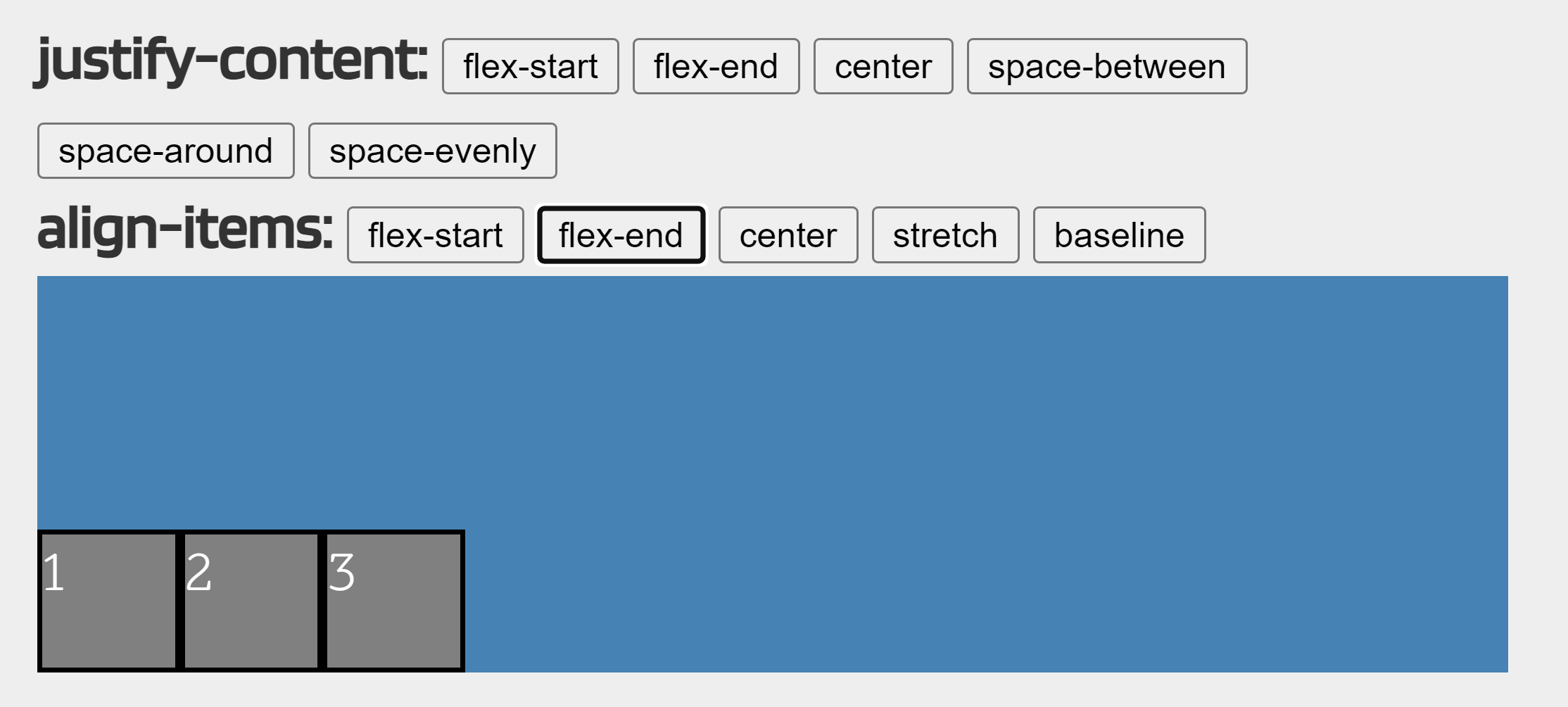
**Sobre el eje secundario**

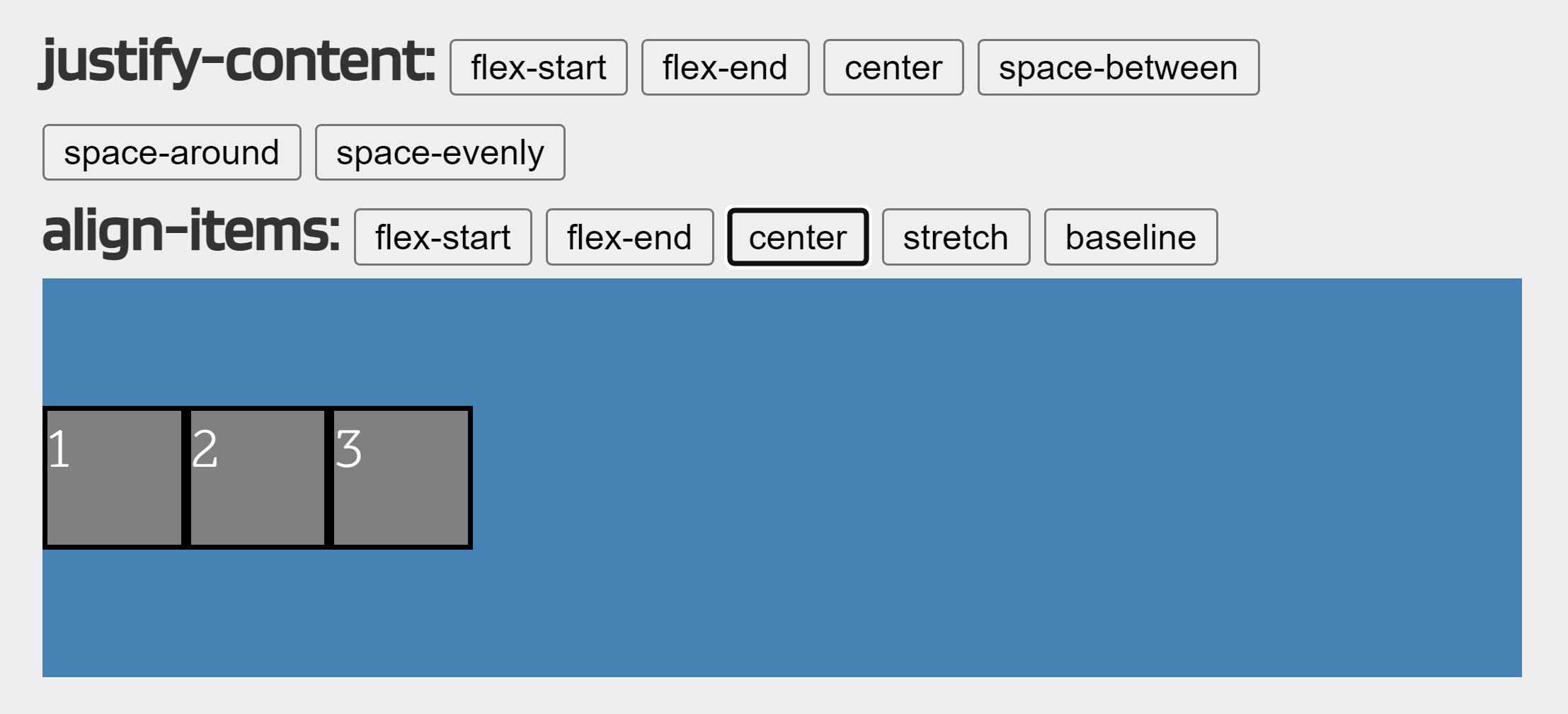
La otra propiedad importante de este apartado es align-items, que se encarga de alinear los ítems en el eje secundario del contenedor. Hay que tener cuidado de no confundir align-content con align-items, puesto que el primero actúa sobre cada una de las líneas de un contenedor multilínea (no tiene efecto sobre contenedores de una sola línea), mientras que align-items lo hace sobre la línea actual. Los valores que puede tomar son los siguientes:

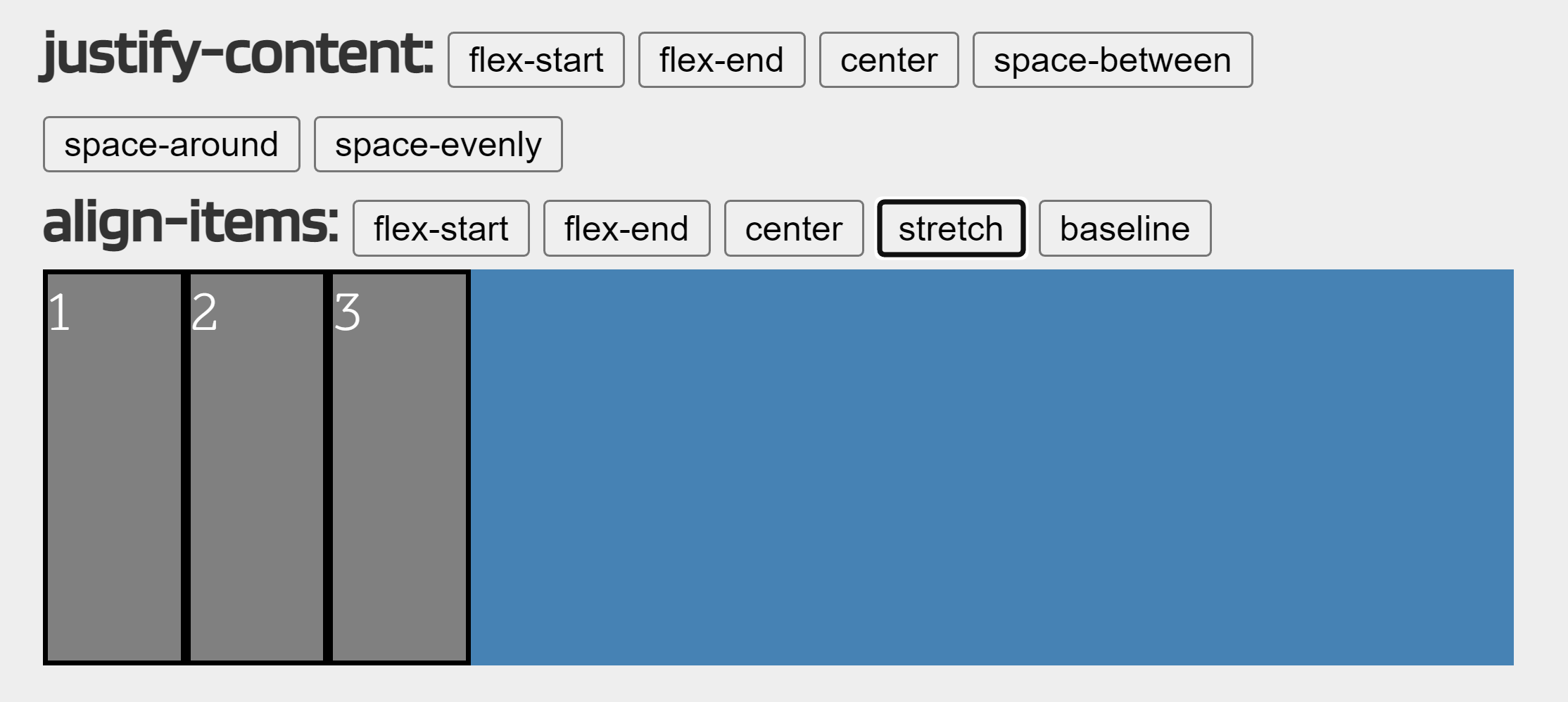
|  |  |
| --- | --- |
| Valor | Descripción |
| flex-start | Alinea los ítems al principio del eje secundario. |
| flex-end | Alinea los ítems al final del eje secundario. |
| center | Alinea los ítems al centro del eje secundario. |
| ****stretch**** | Alinea los ítems estirándolos de modo que cubran desde el inicio hasta el final del contenedor. |
| baseline | Alinea los ítems en el contenedor según la base del contenido de los ítems del contenedor. |

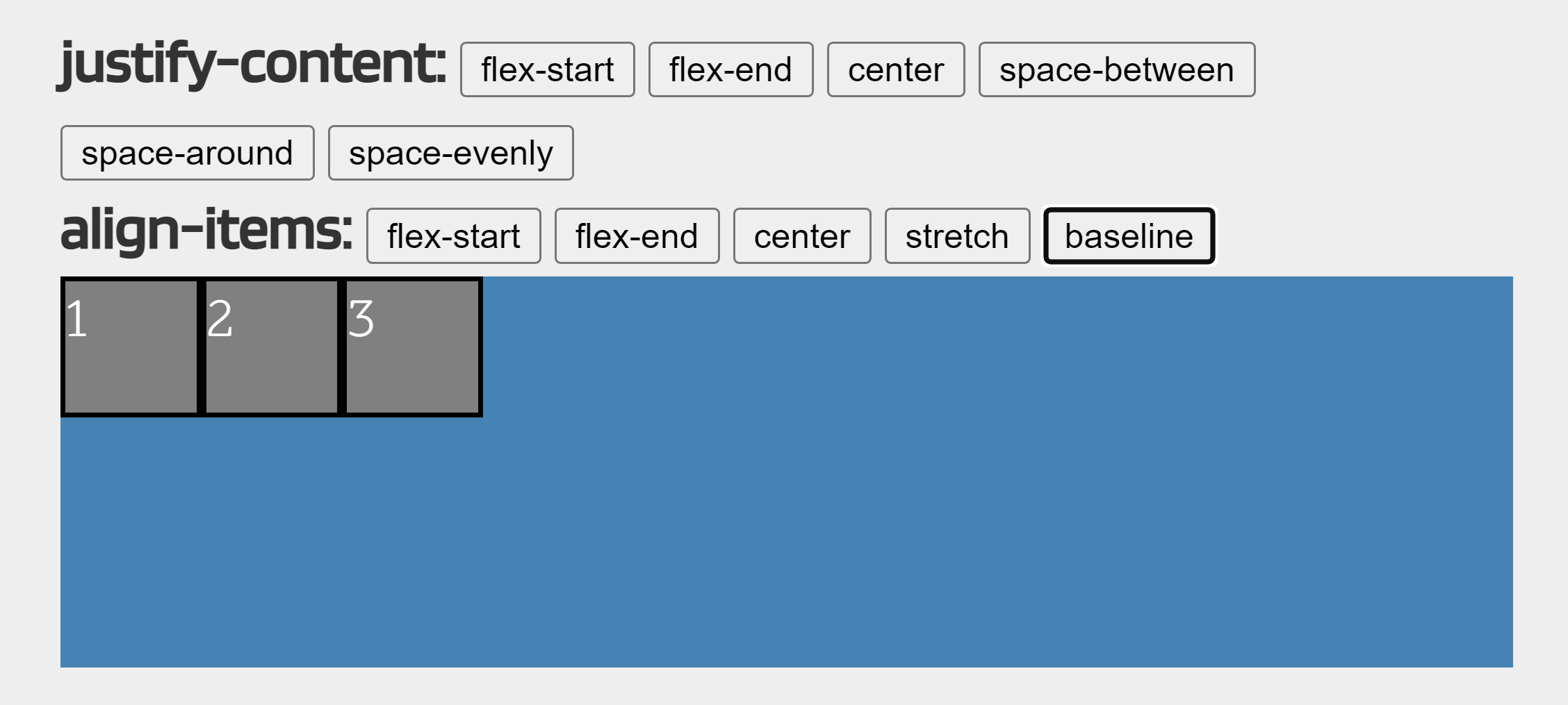
Veamos ejemplos con justify-content y align-items:

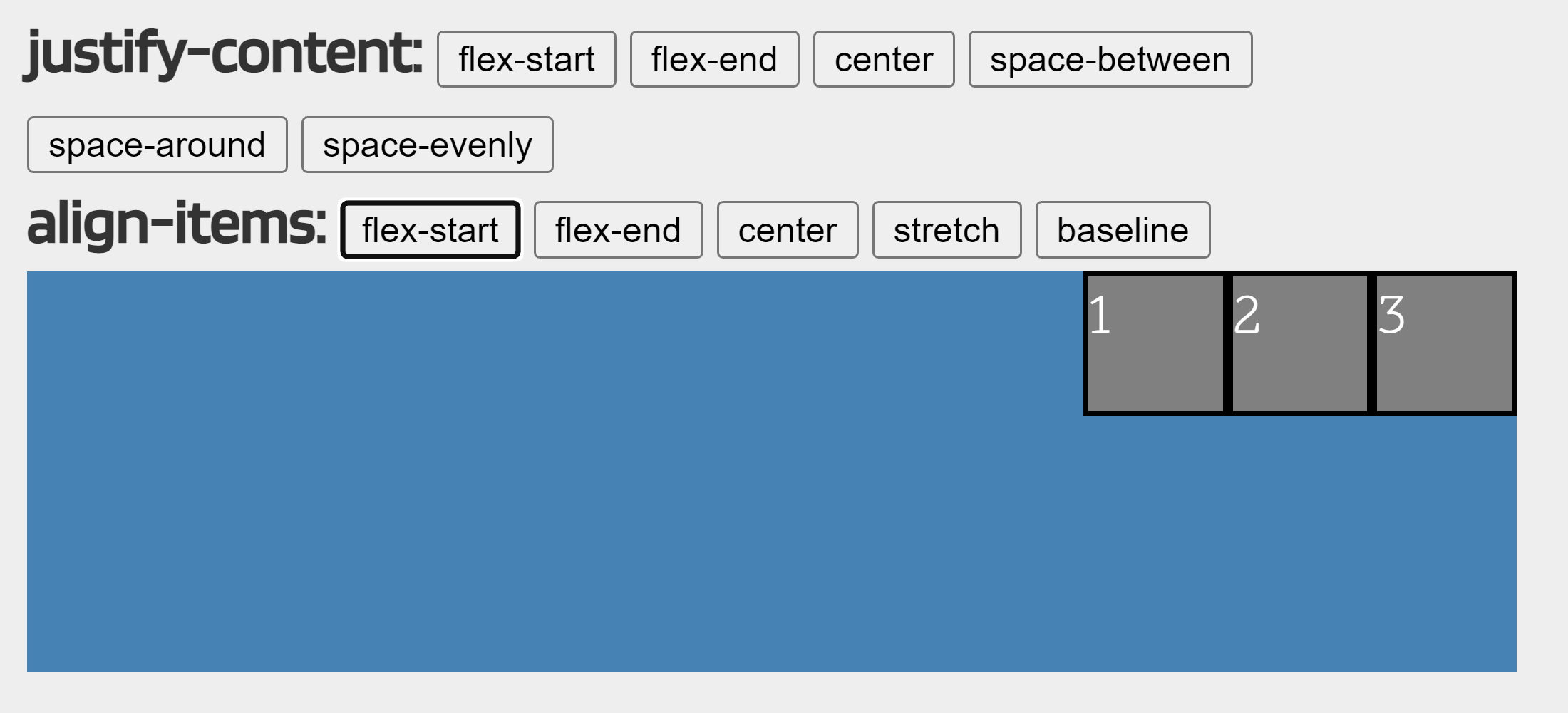
****

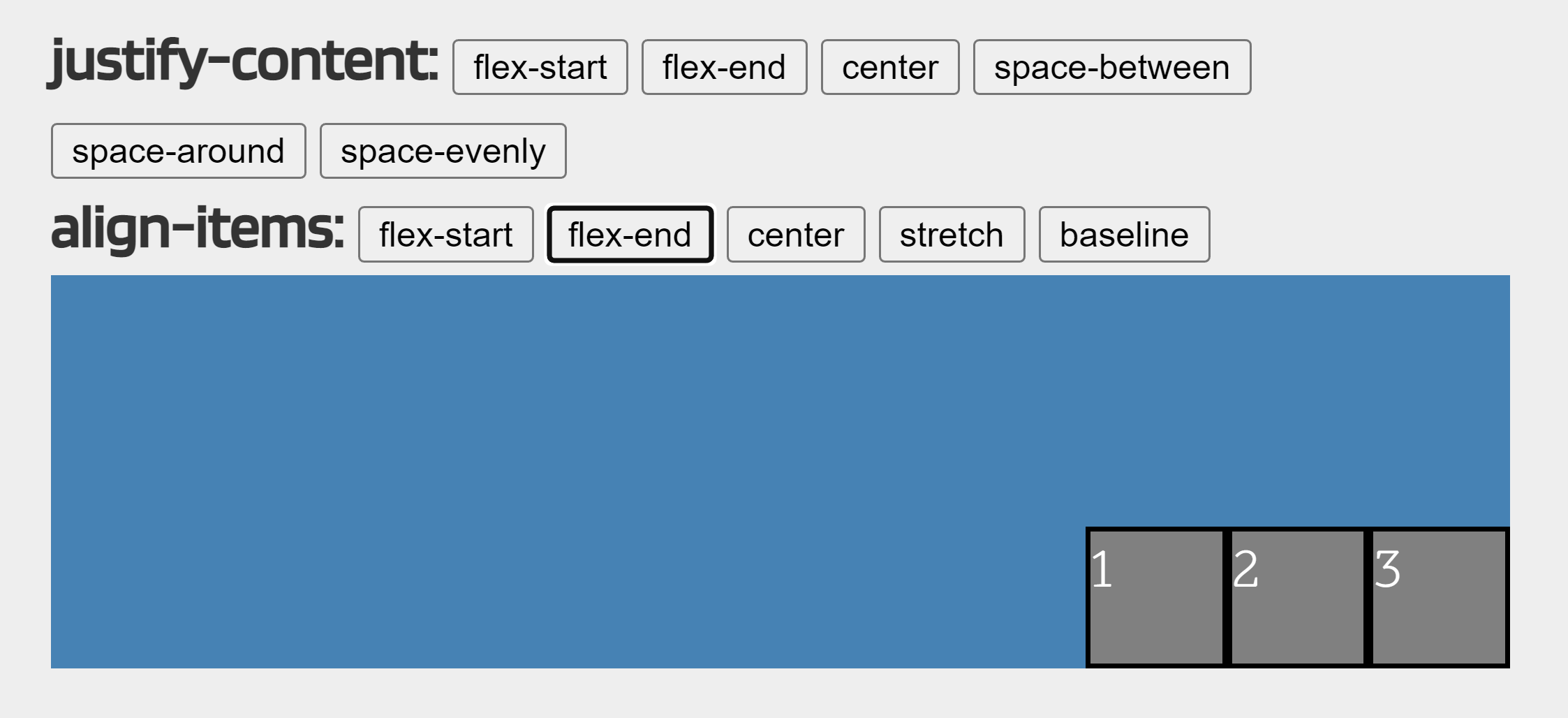
****

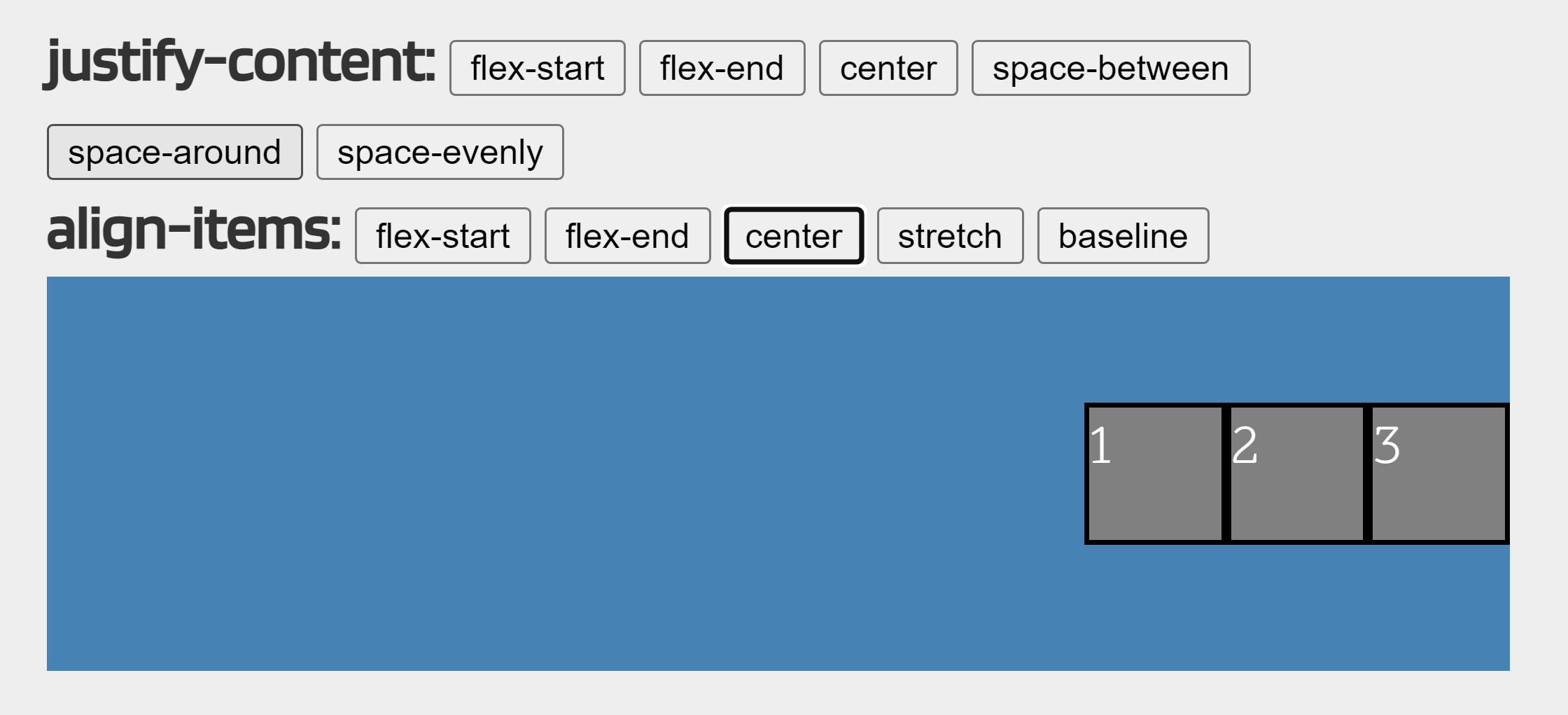
****

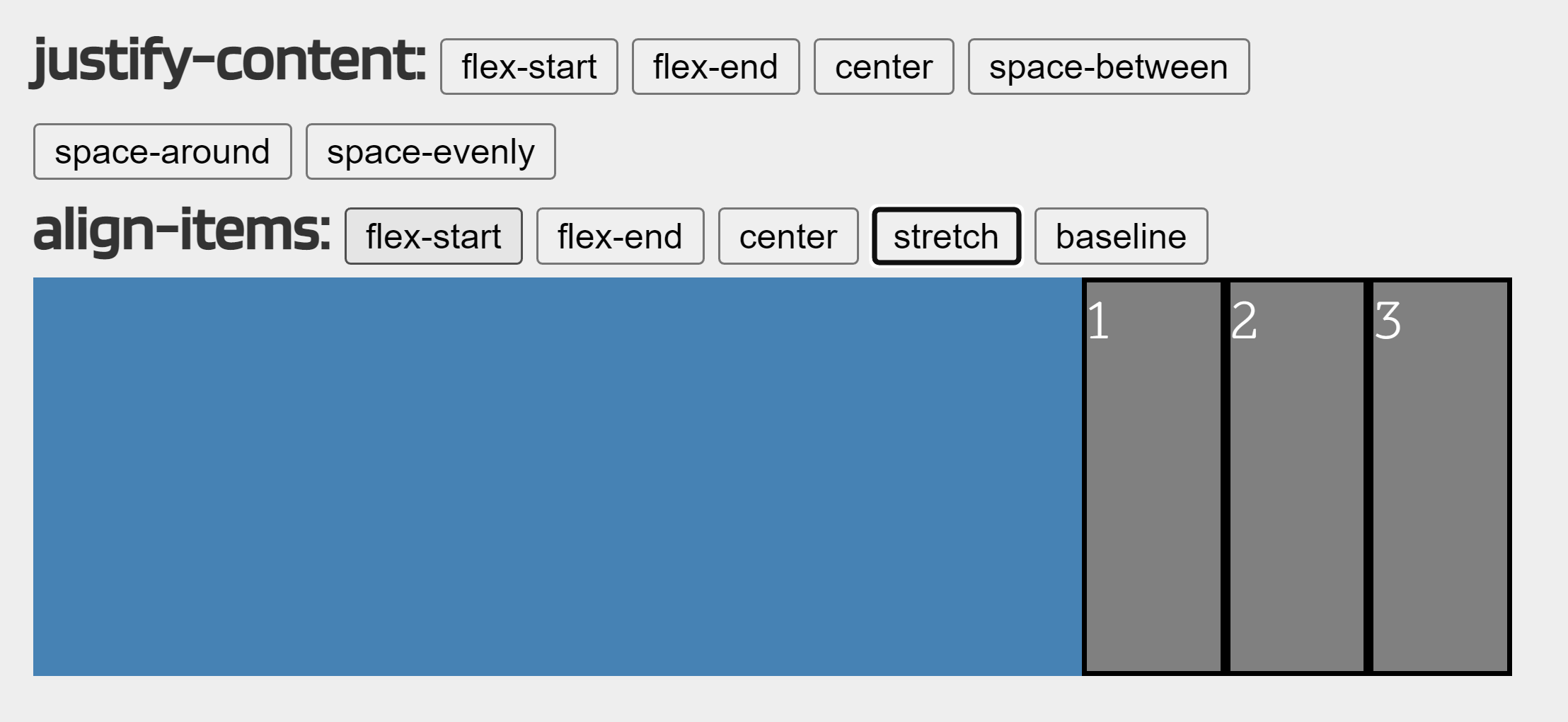
****

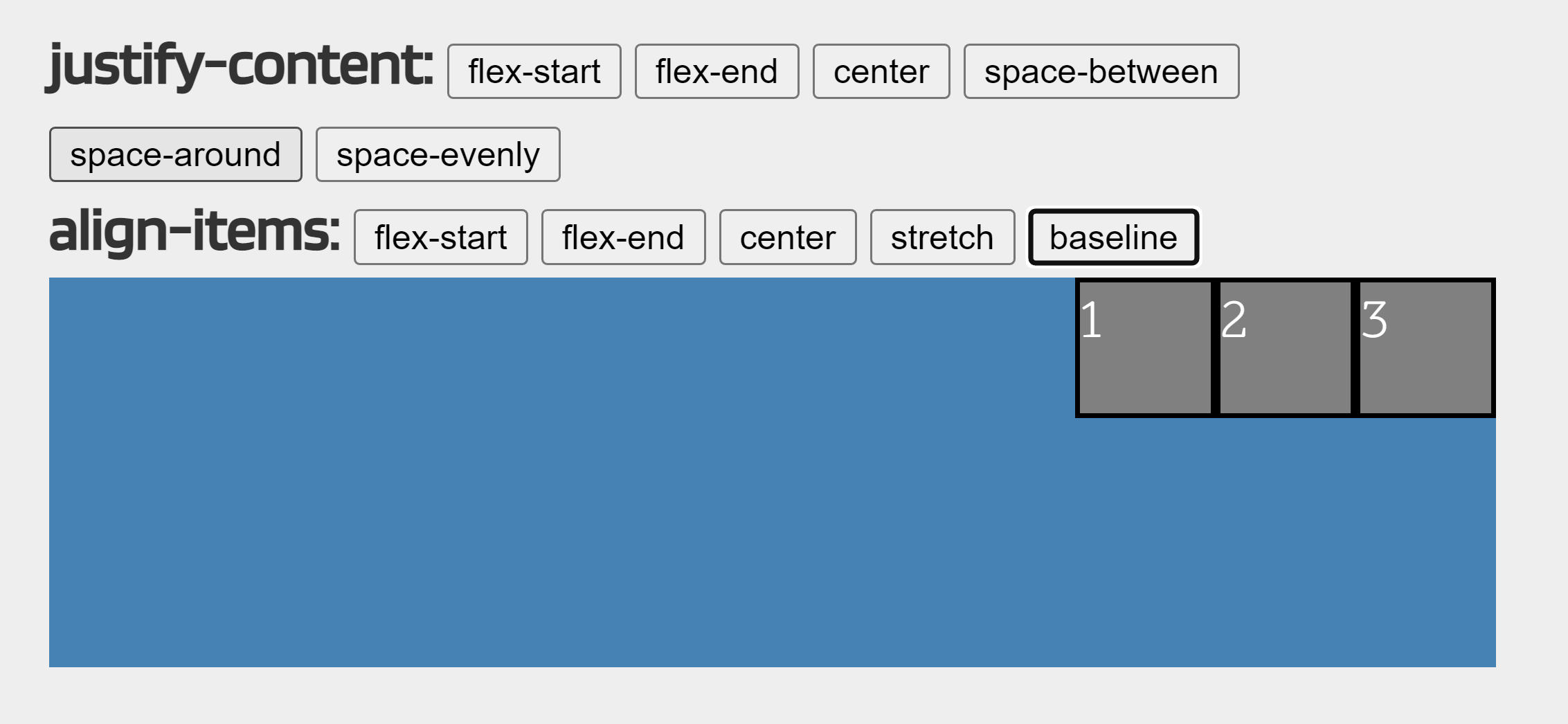
****

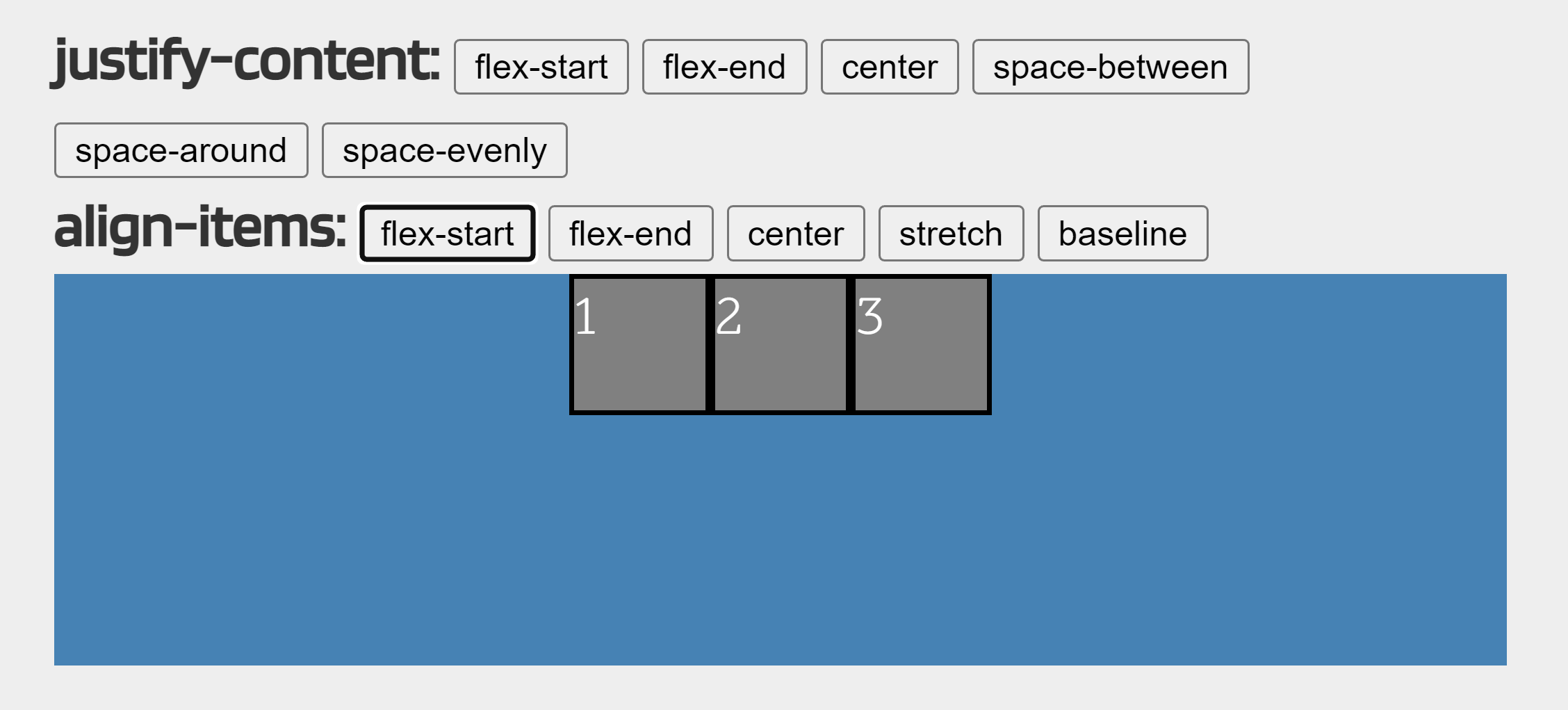
****

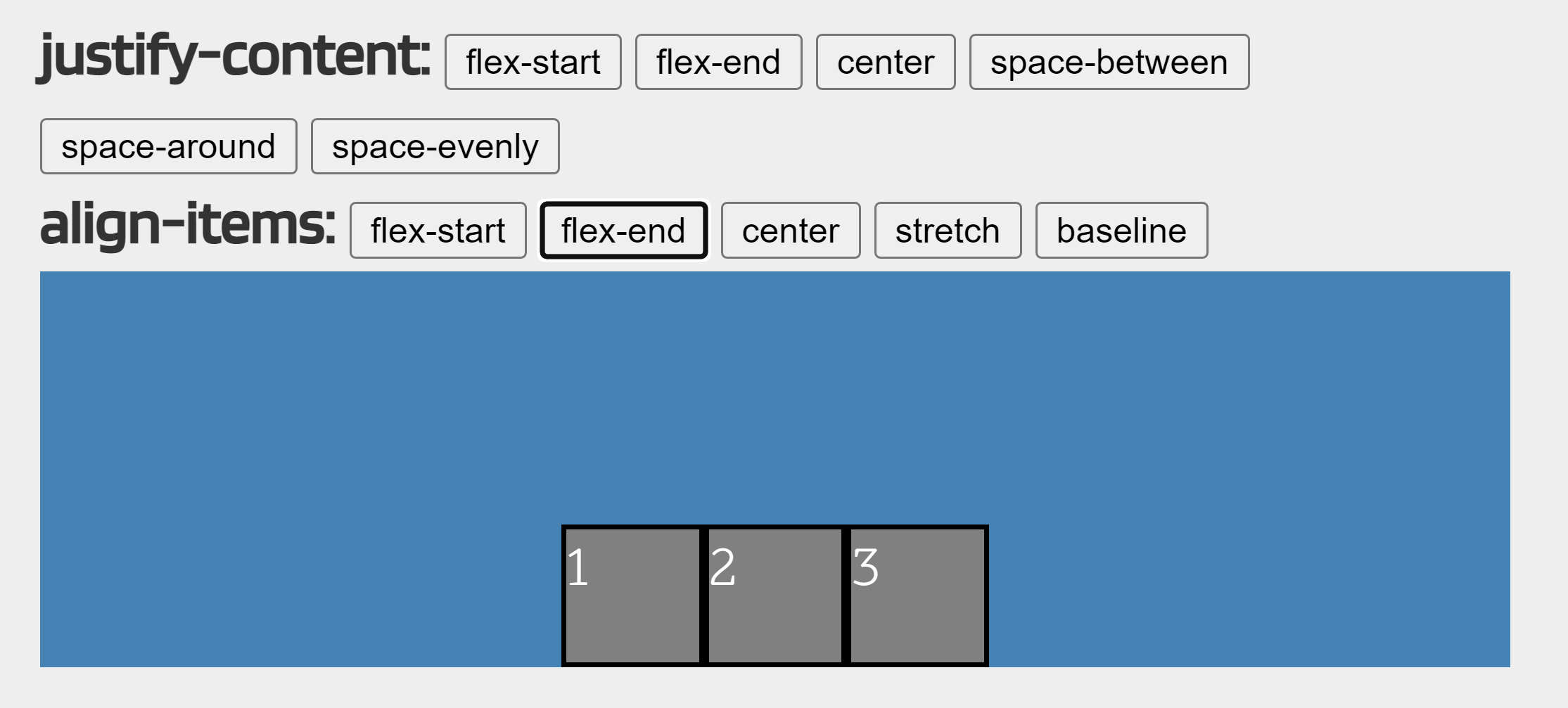
****

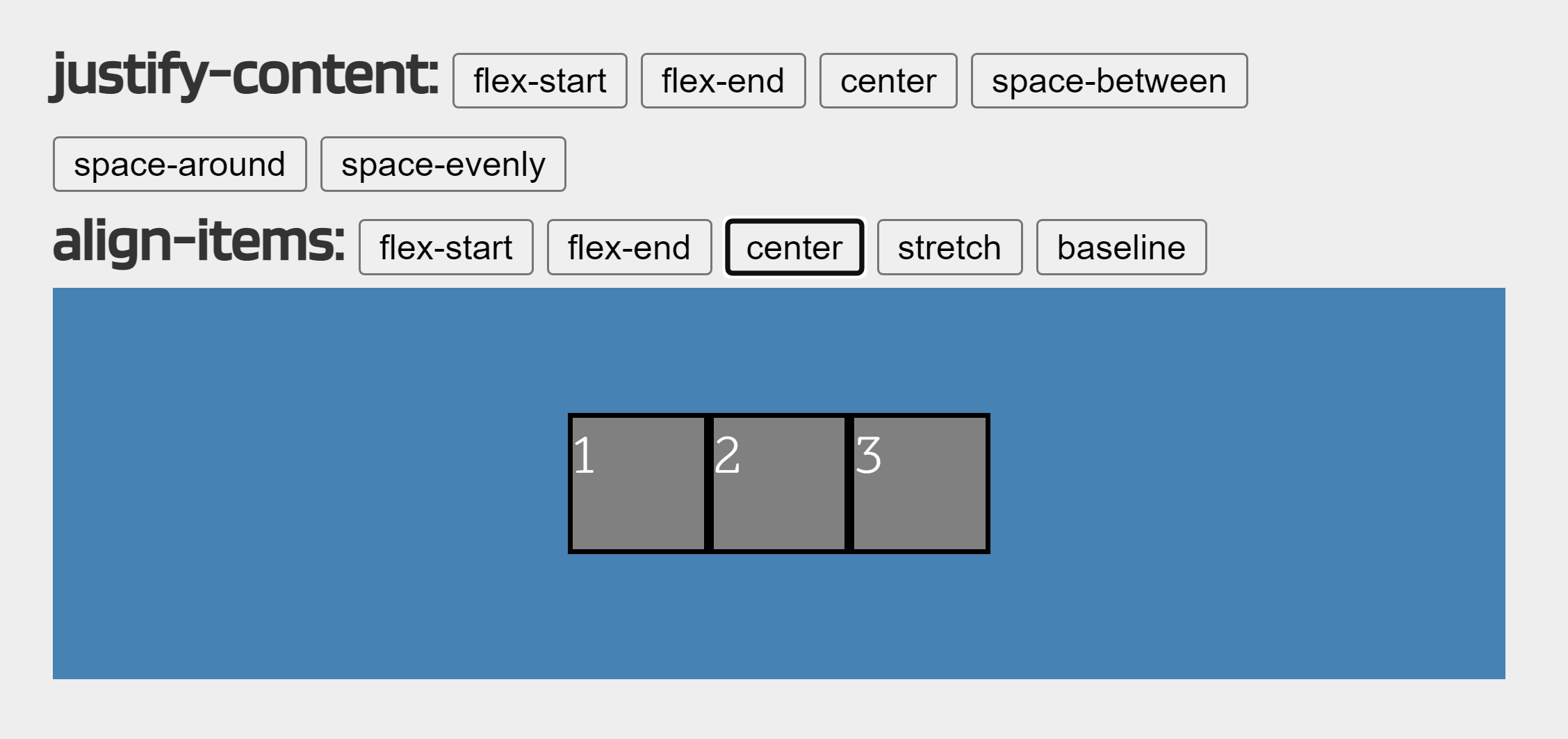
****

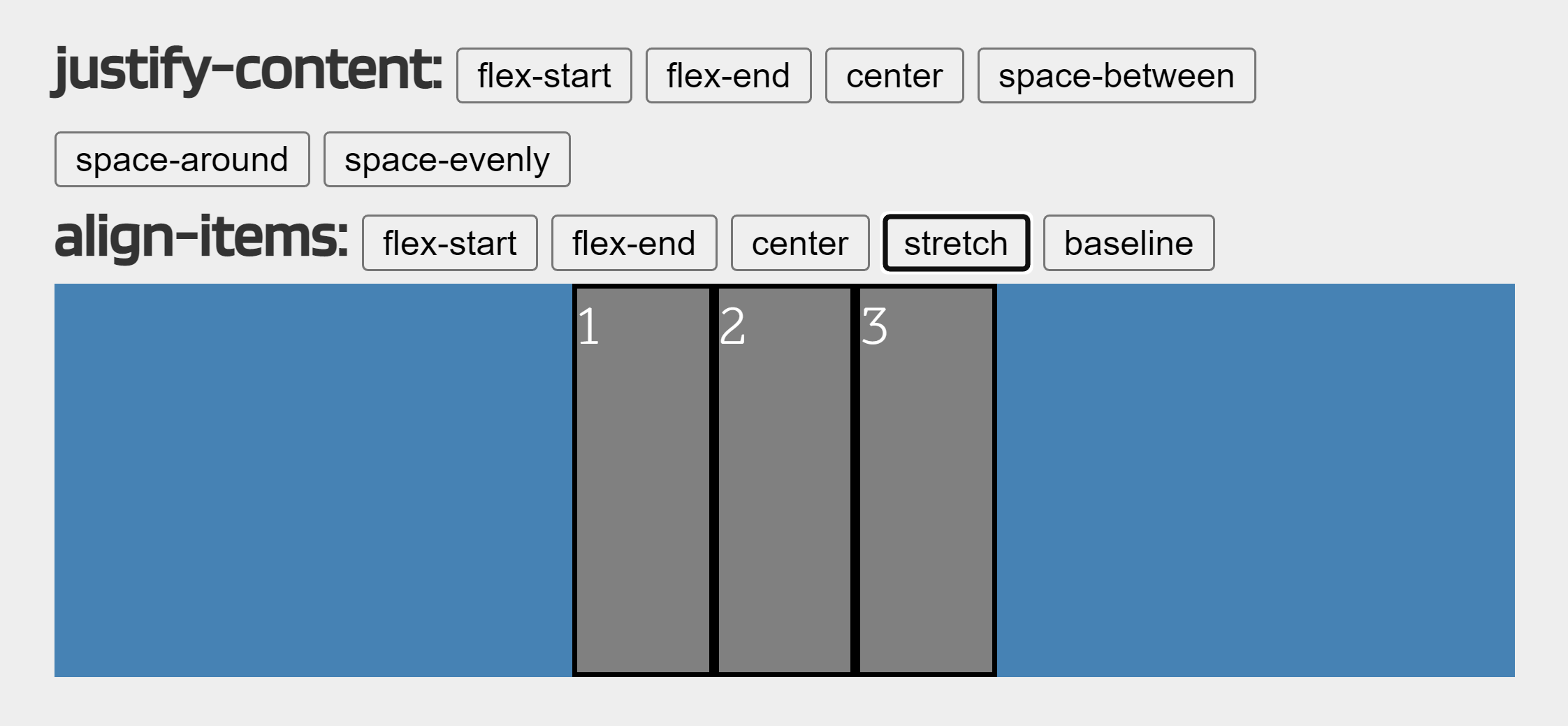
****

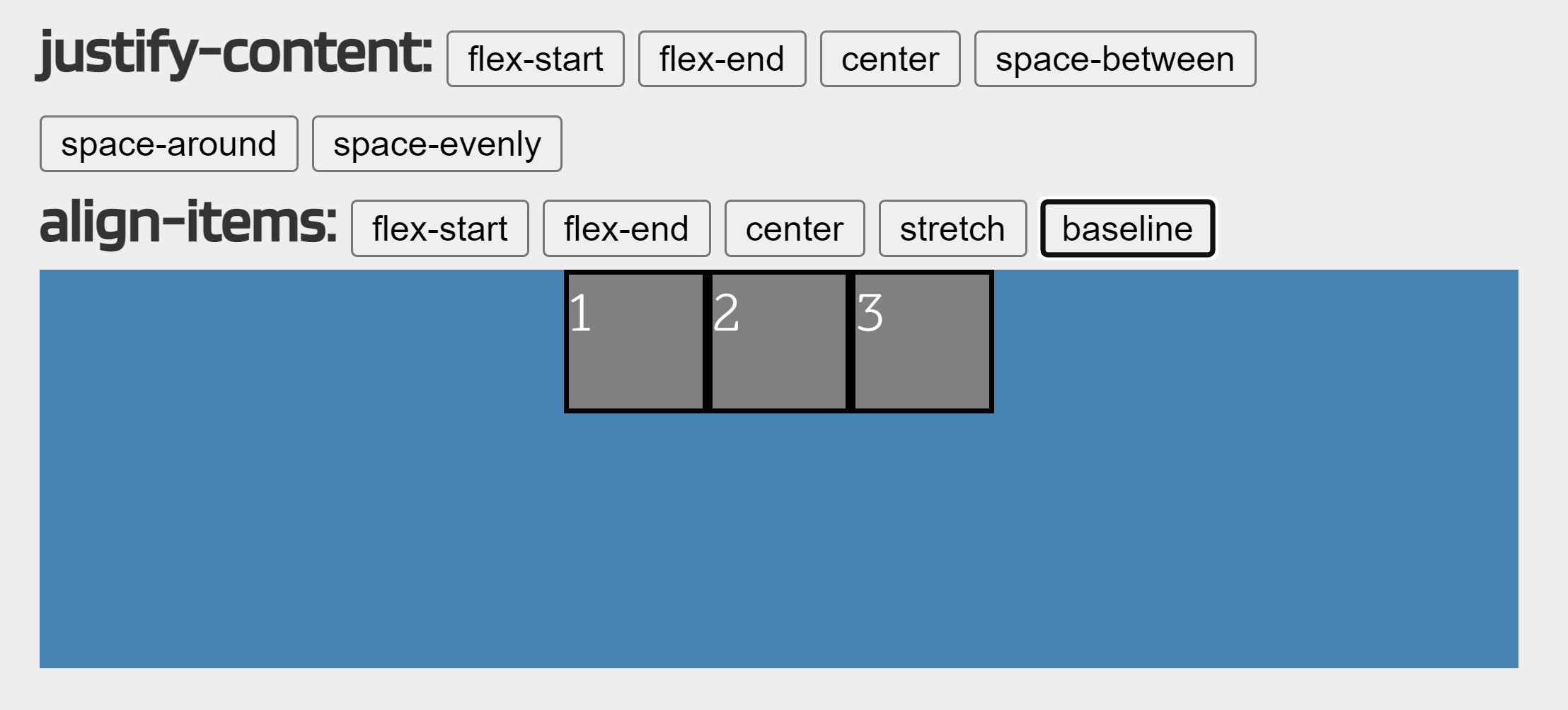
****

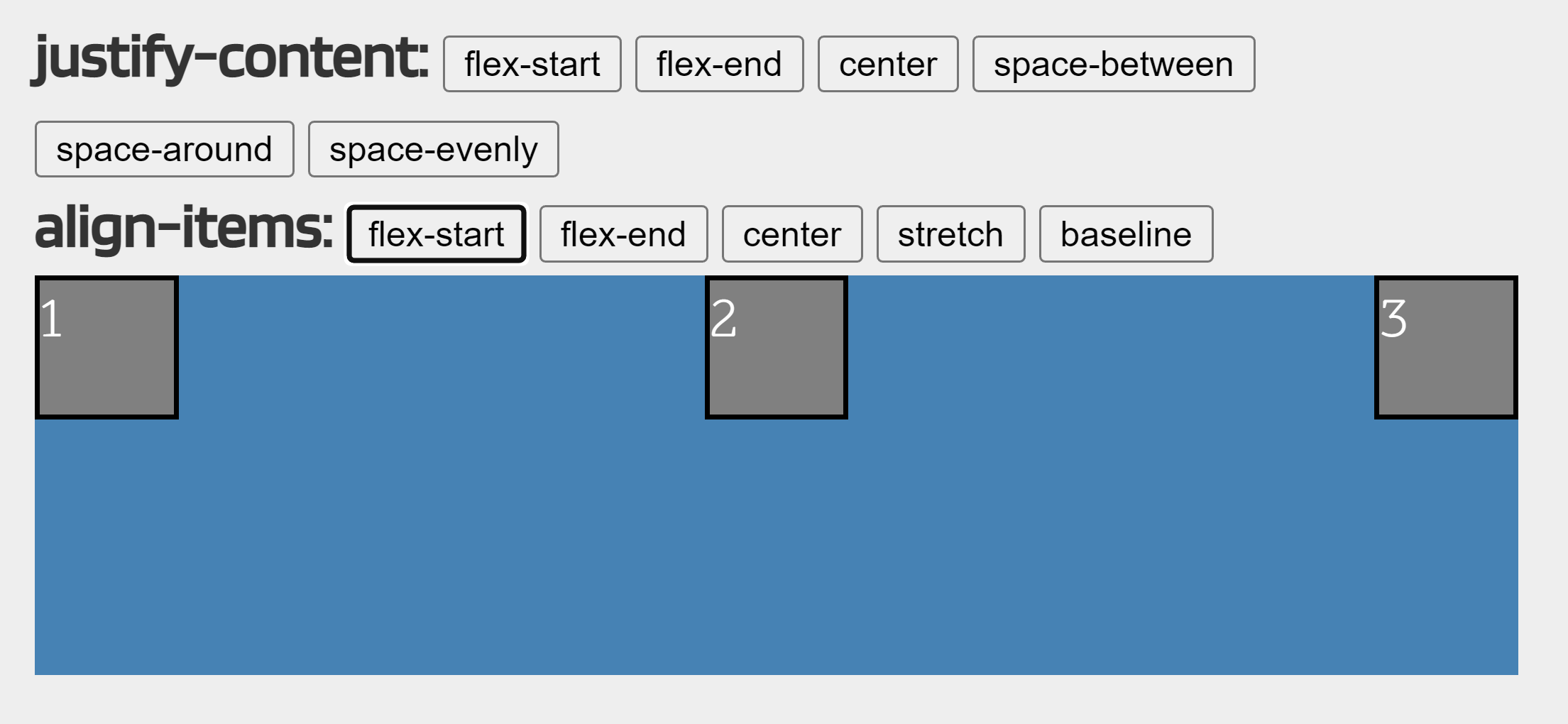
****

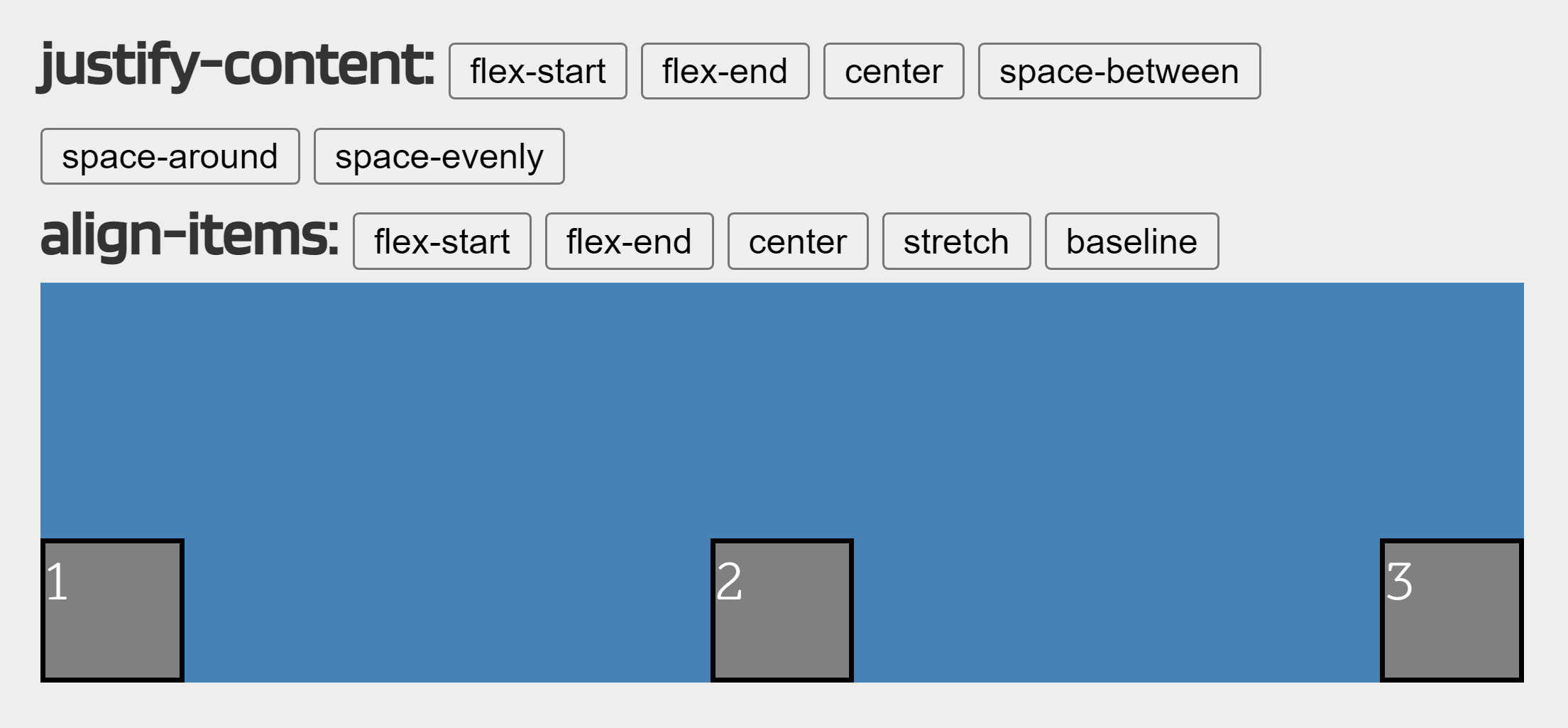
****

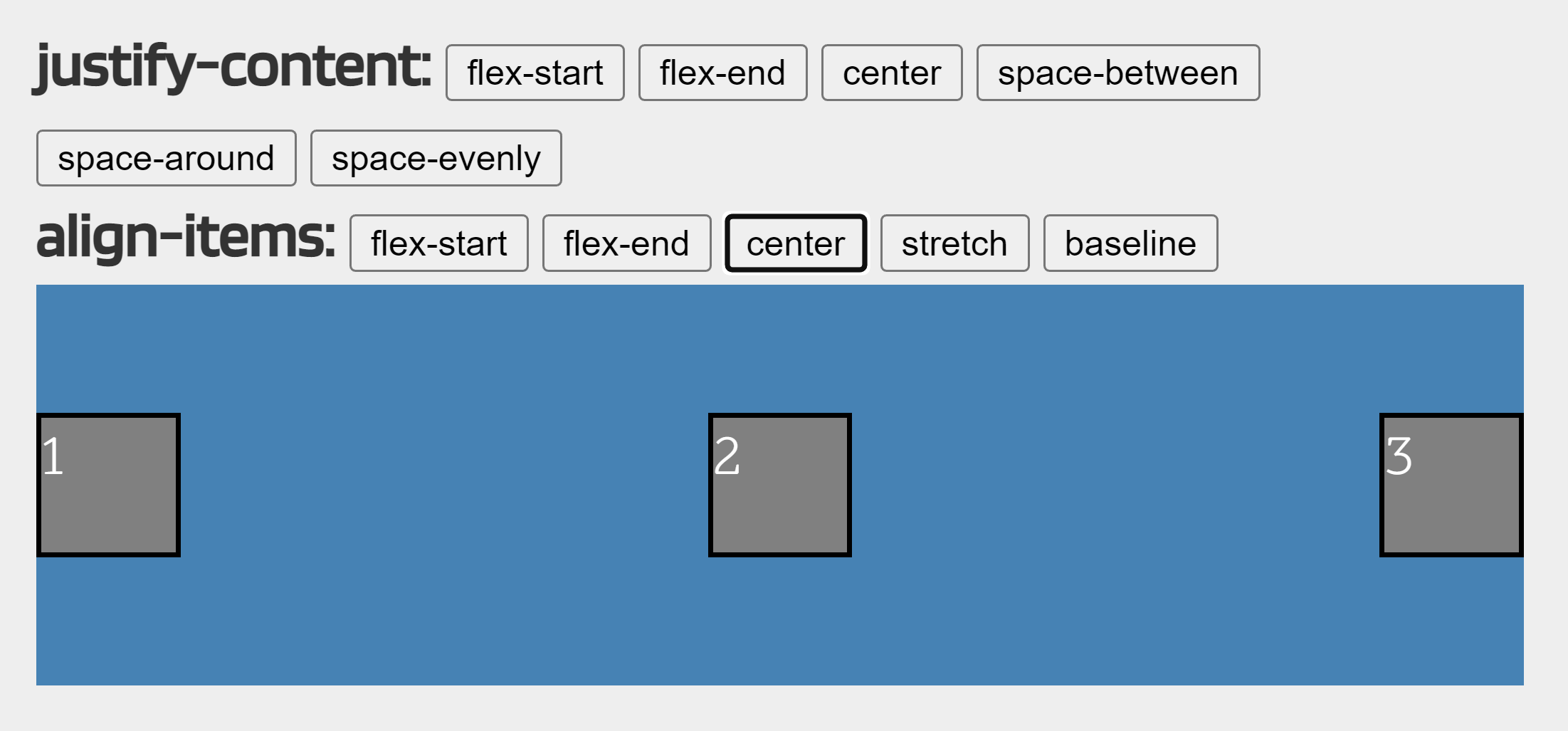
****

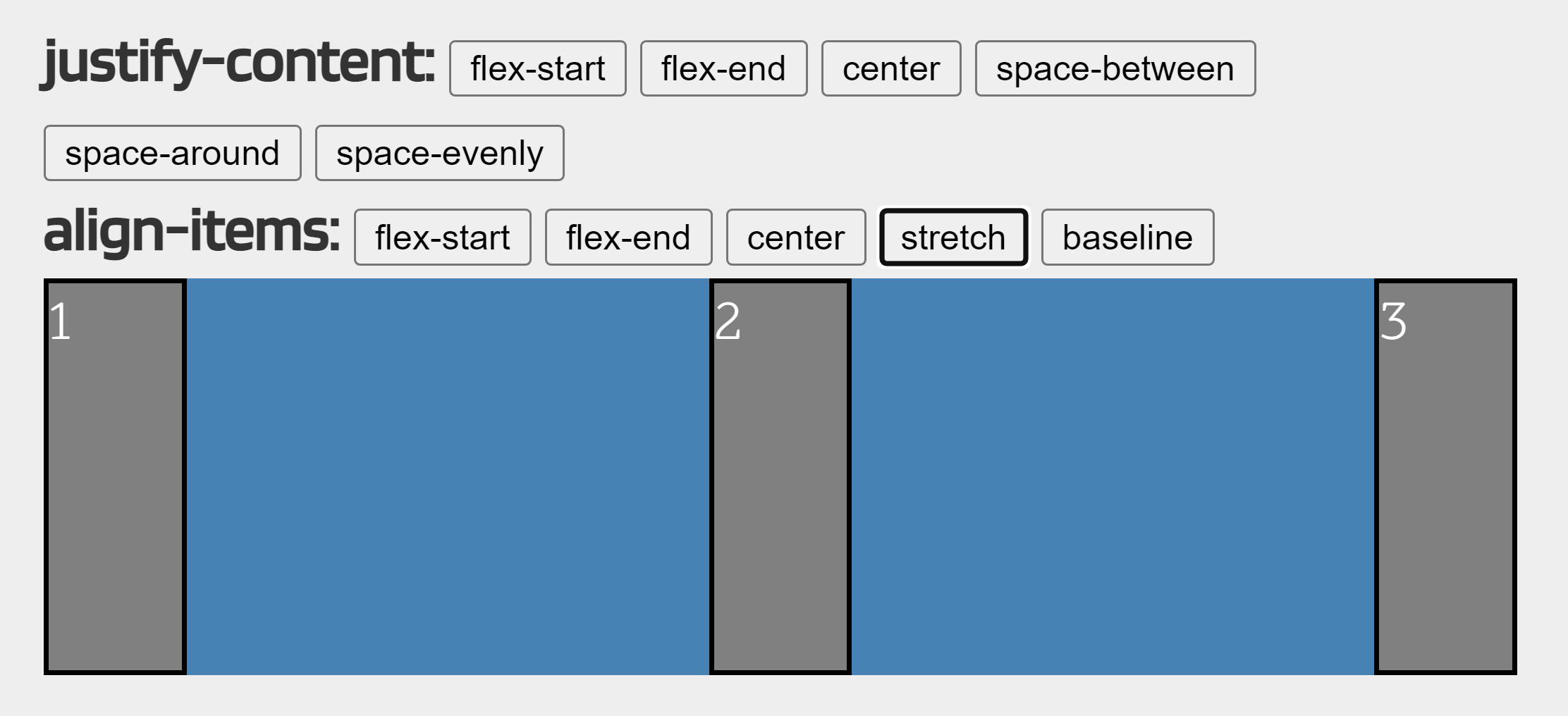
****

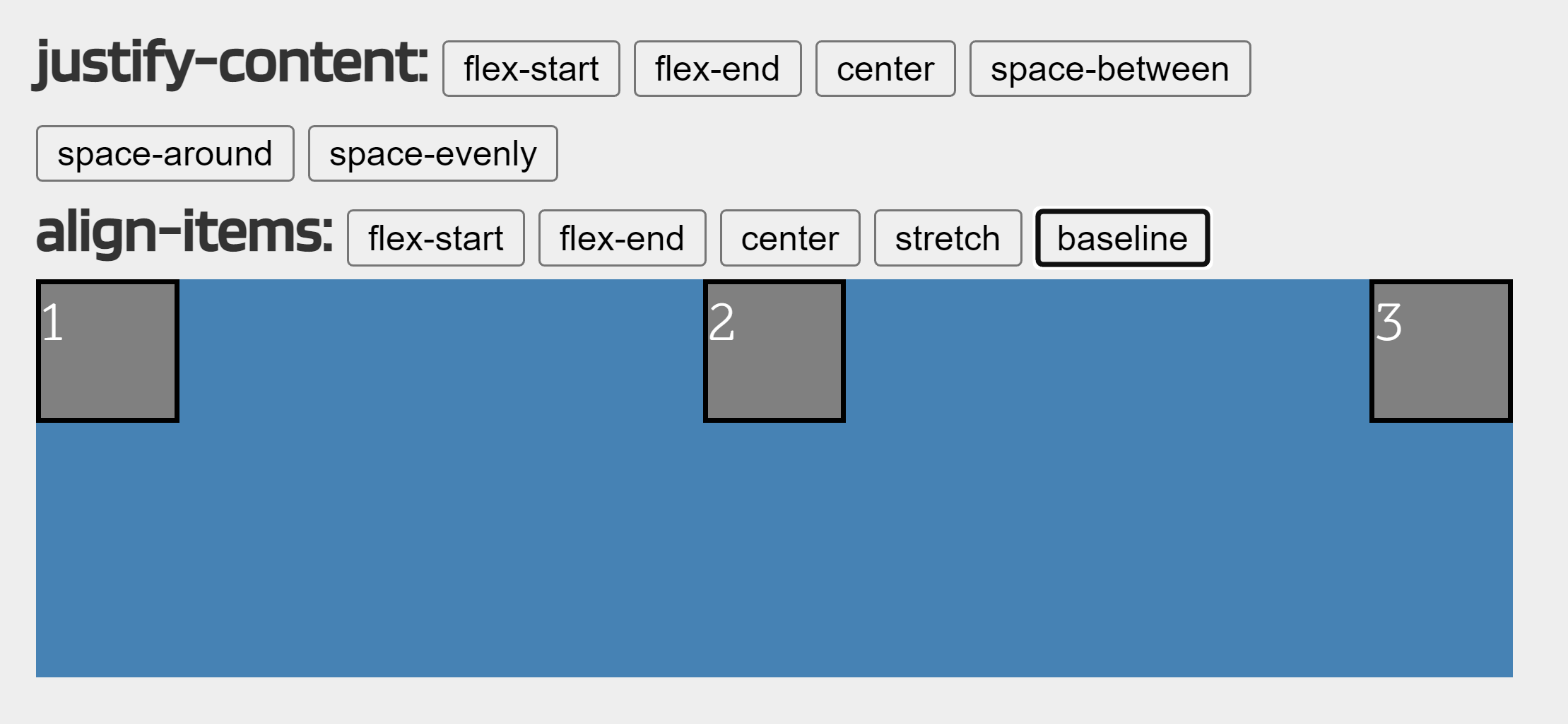
****

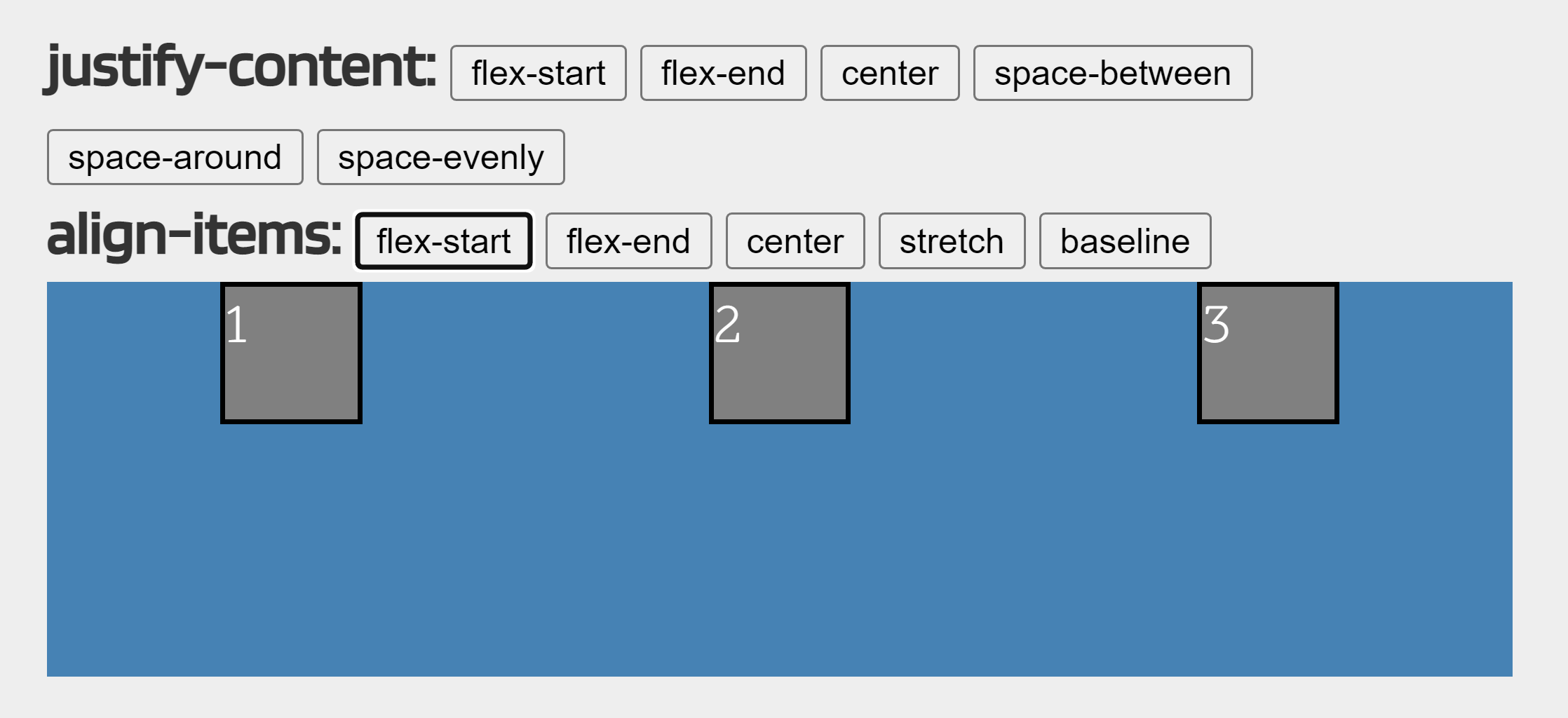
****

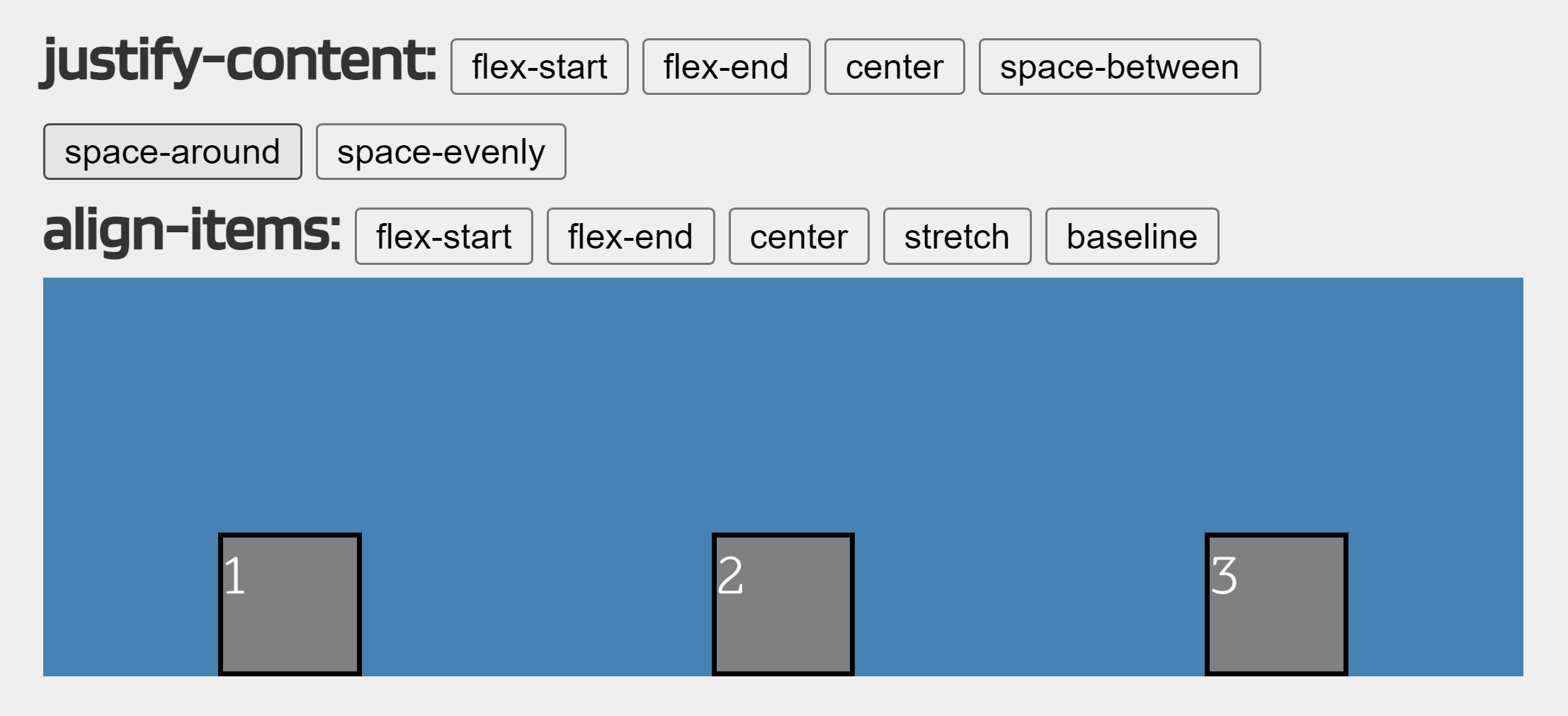
****

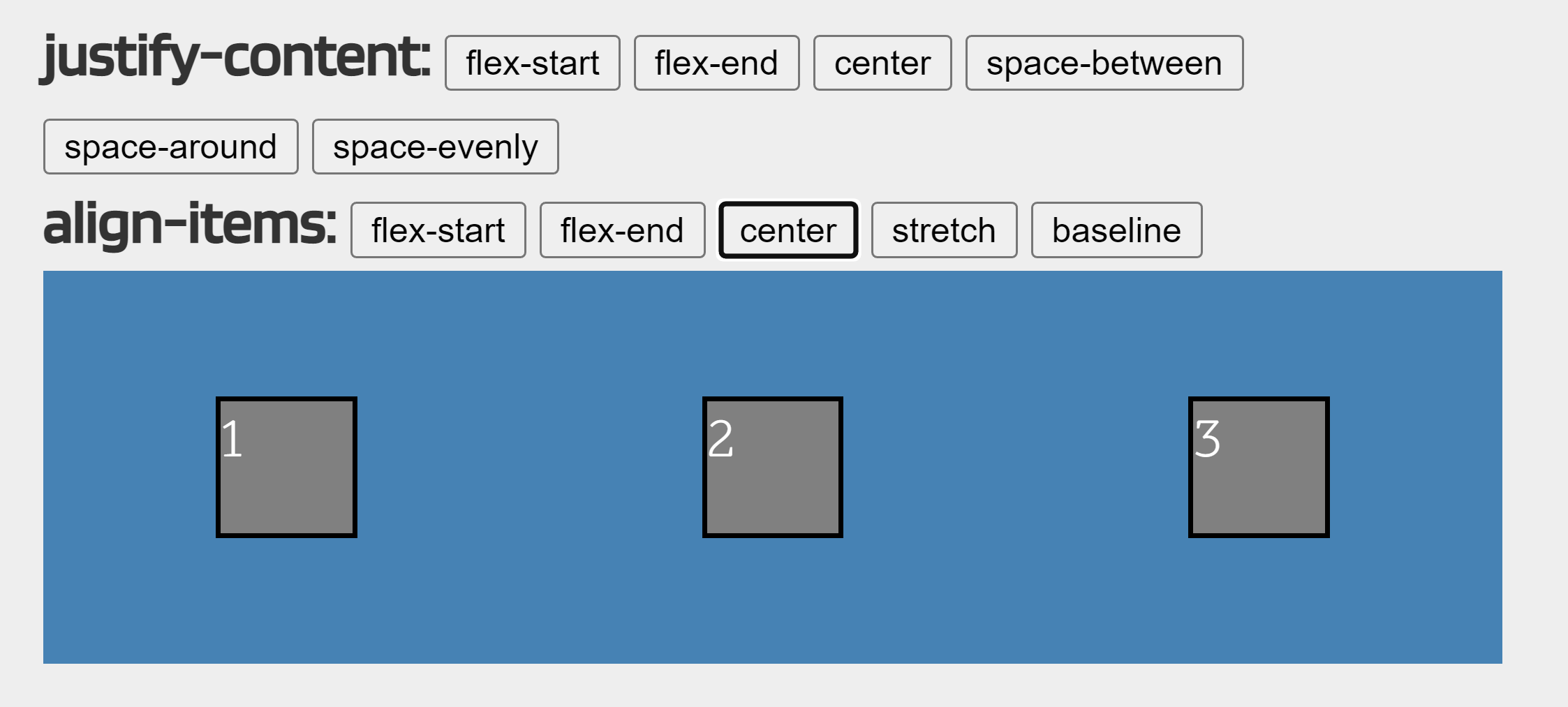
****

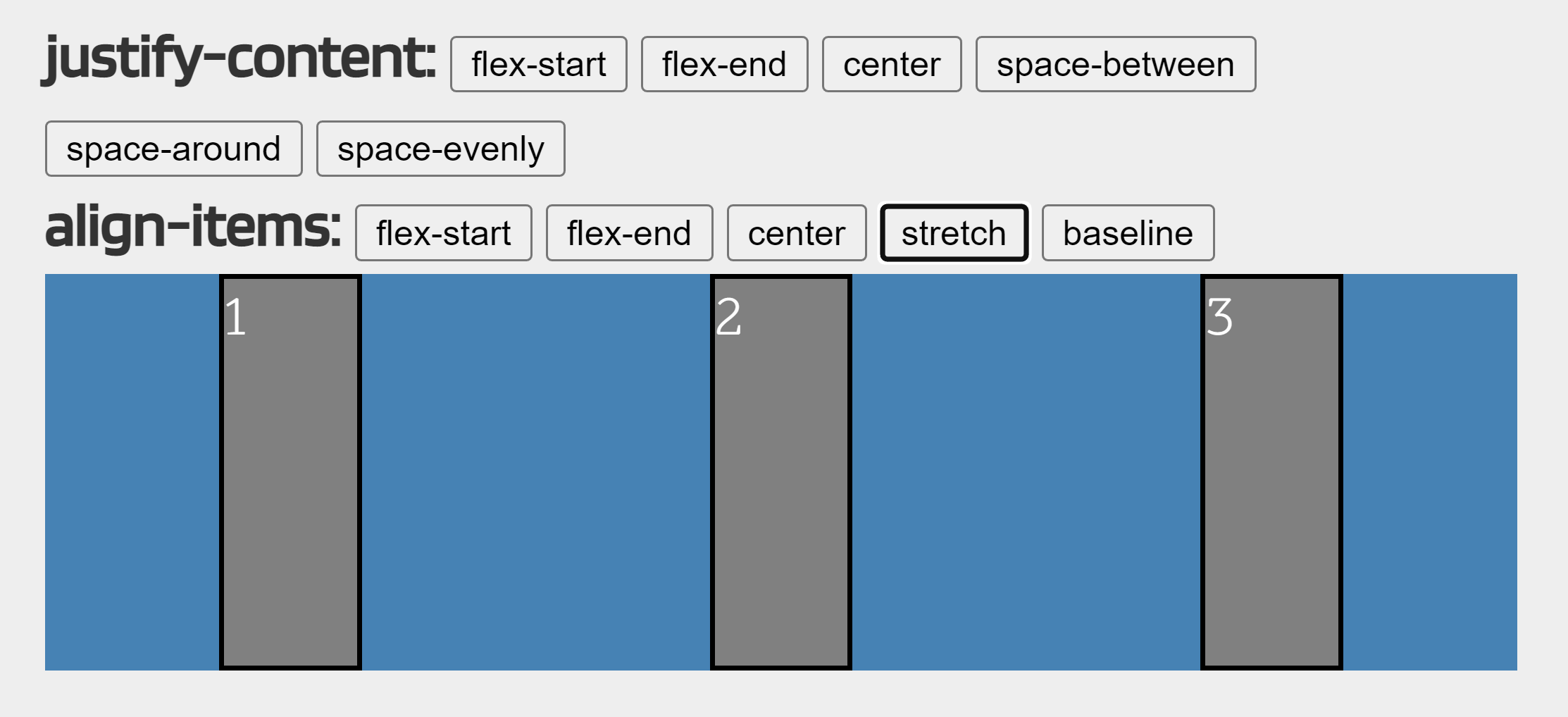
****

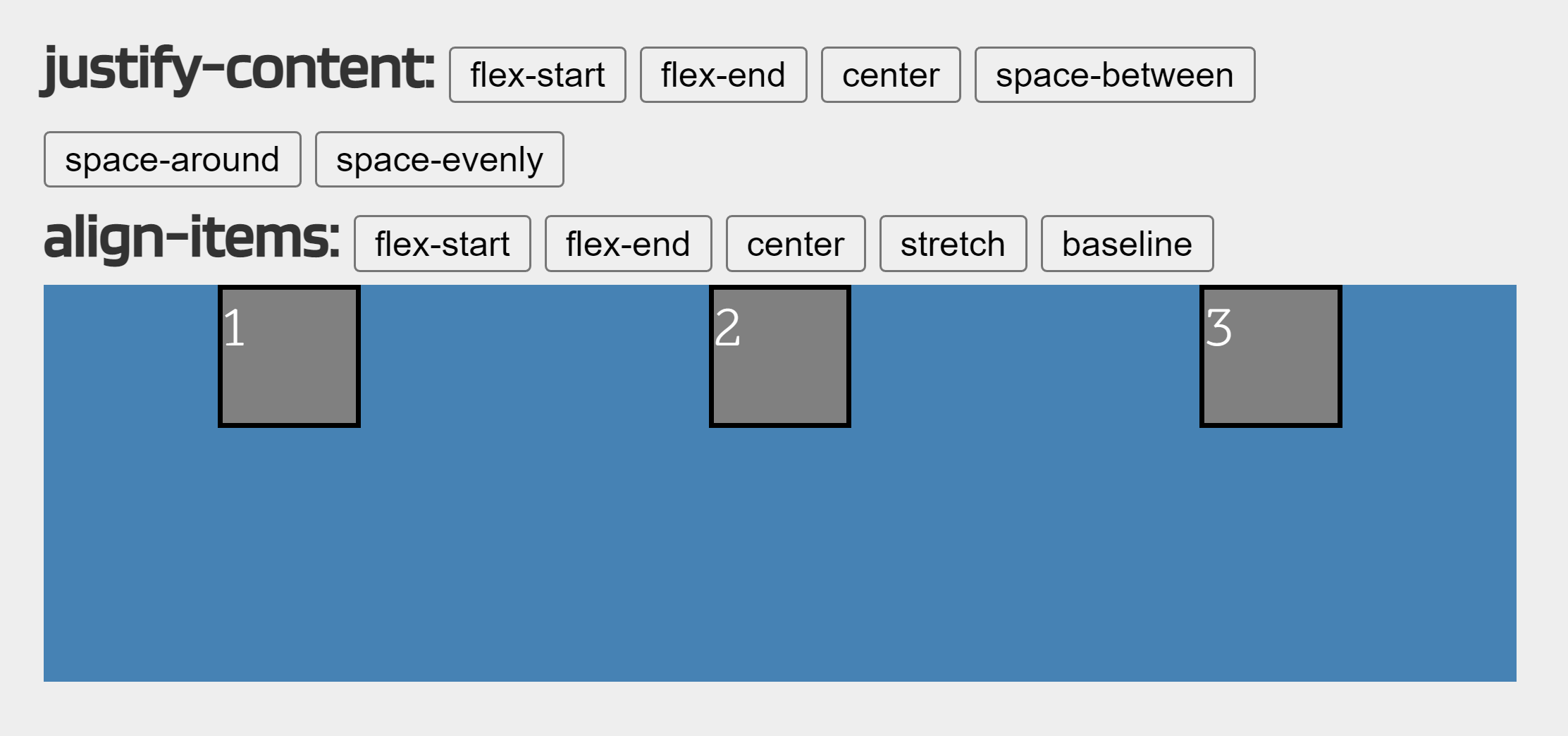
****

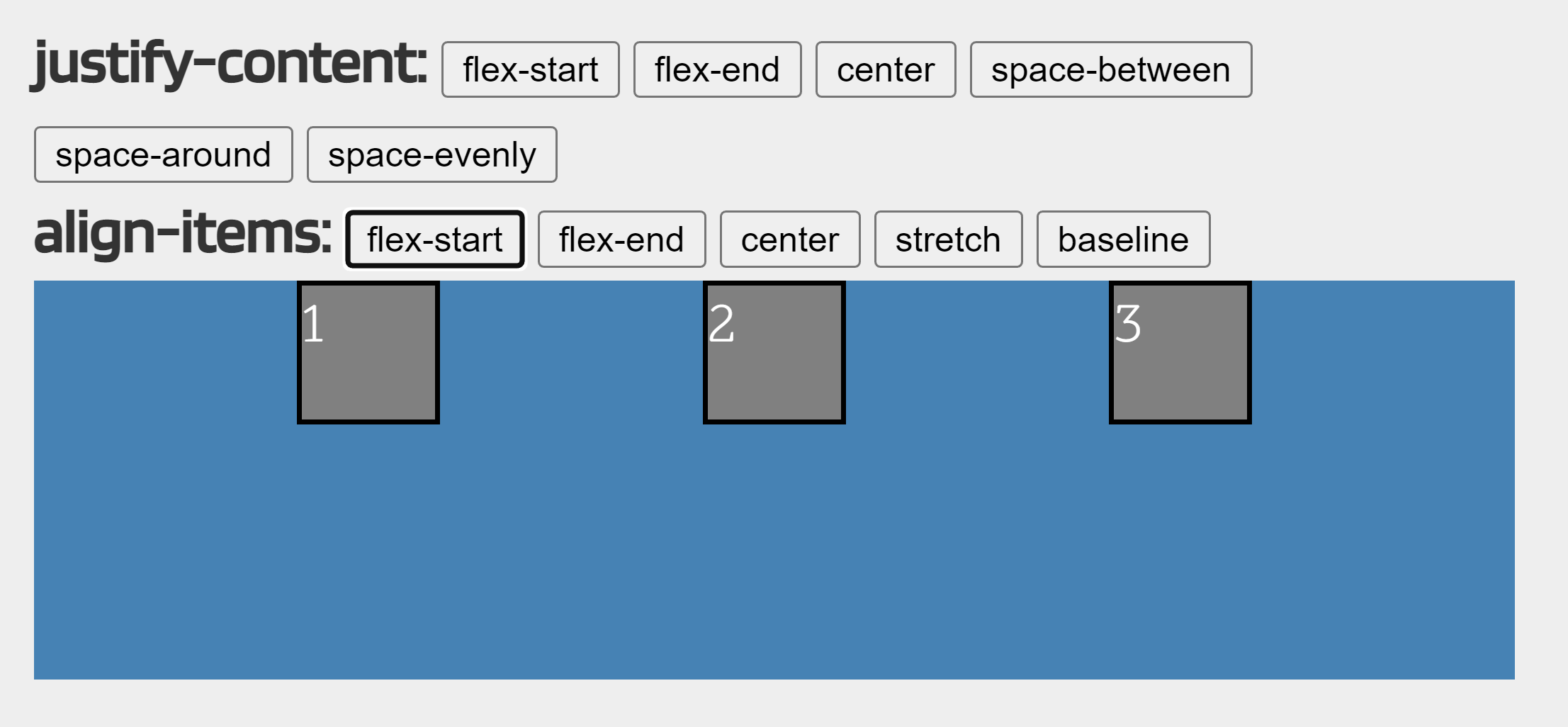
****

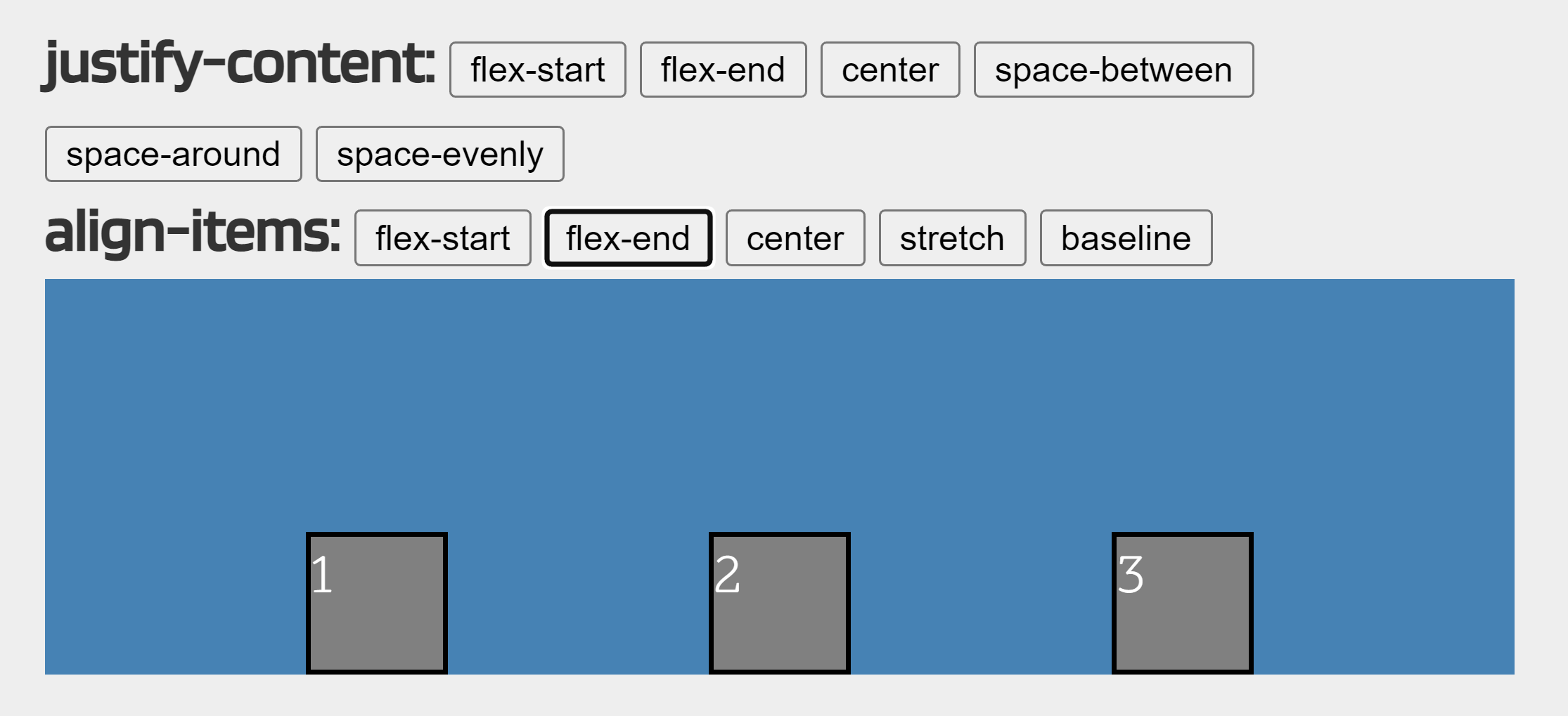
****

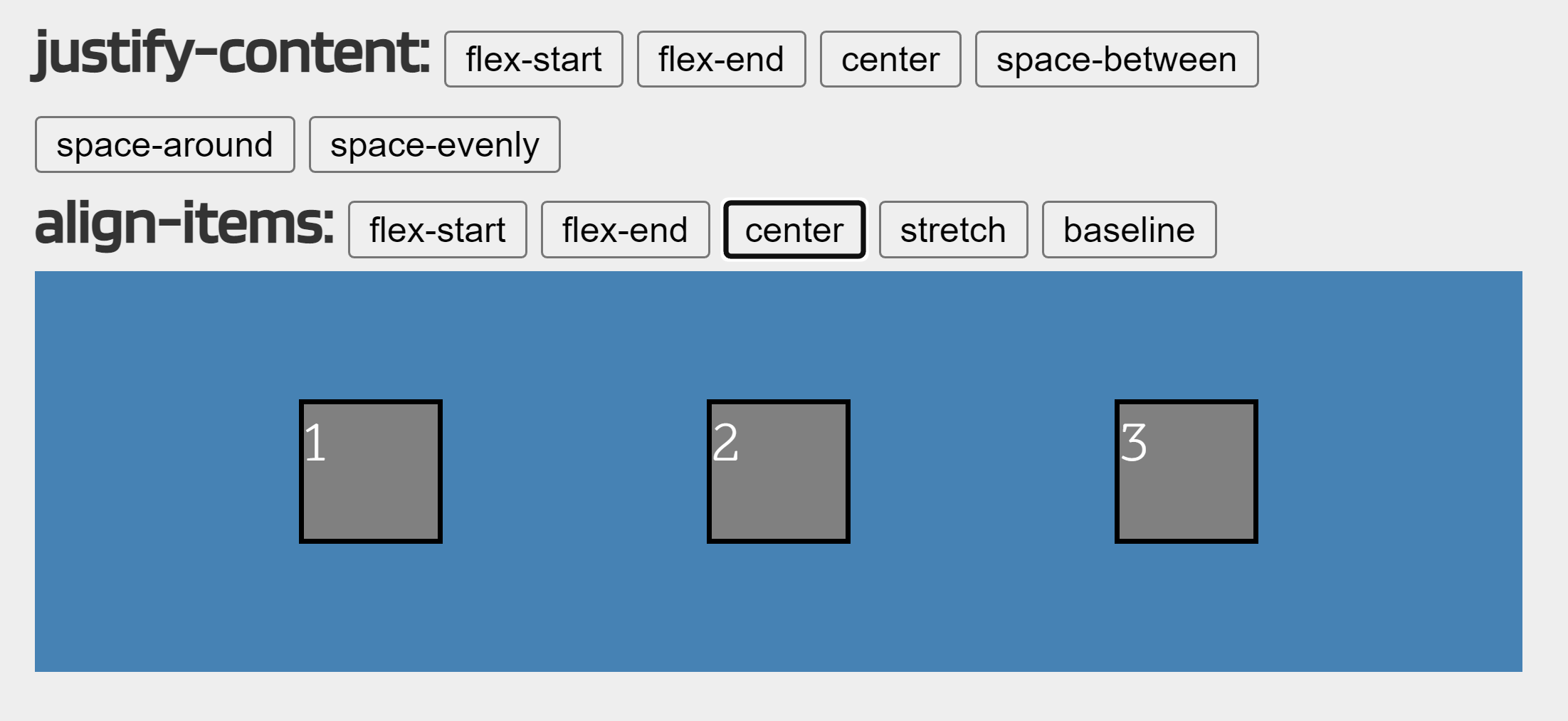
****

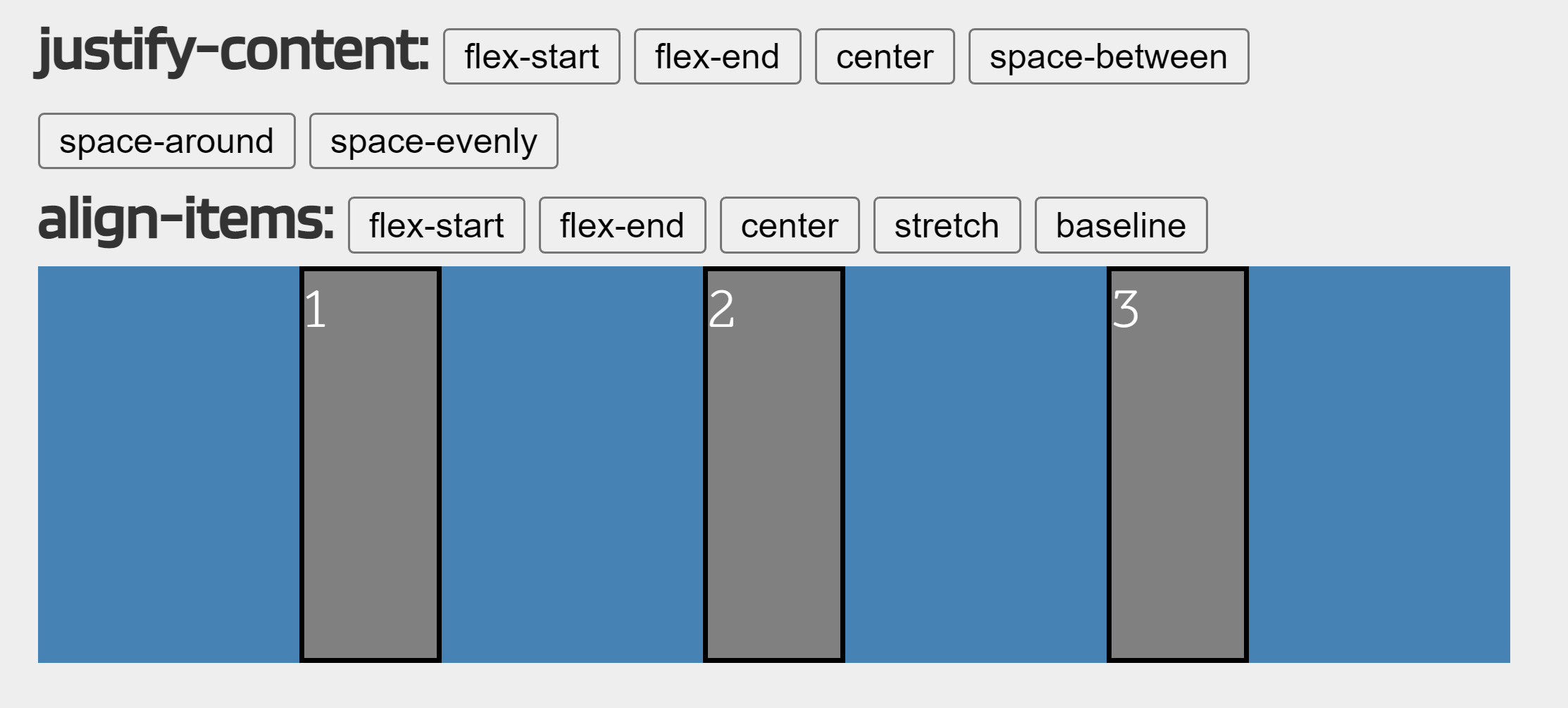
****

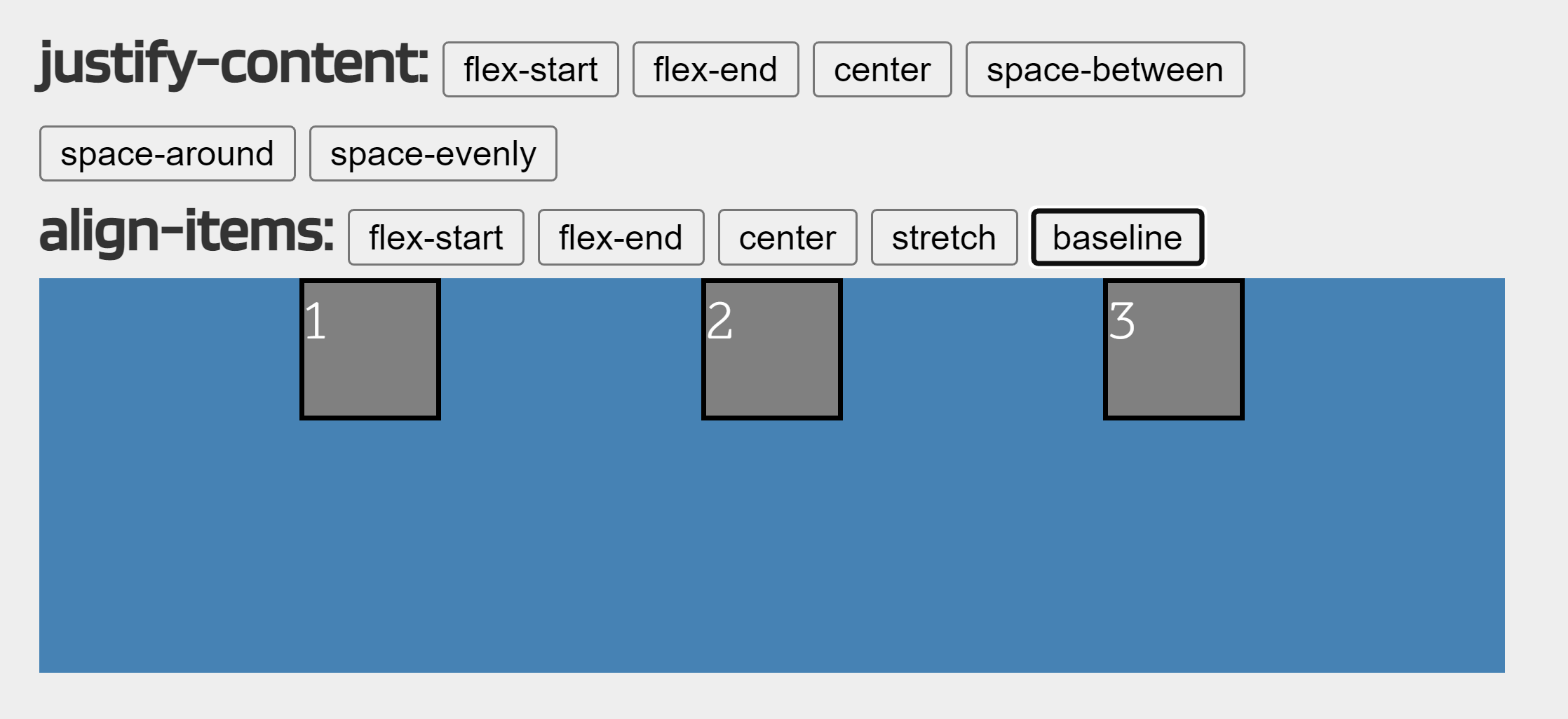
****

****

****

****

****

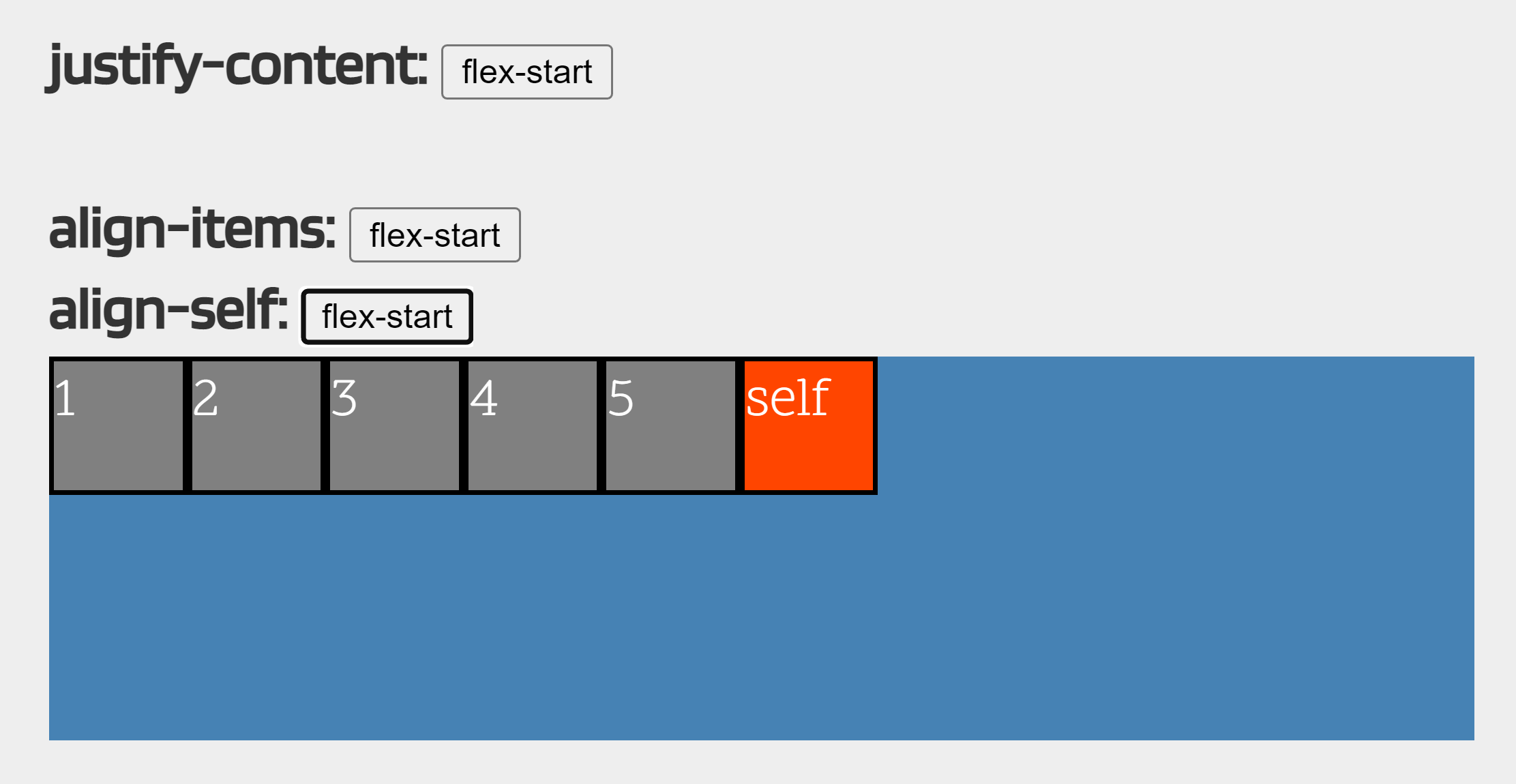
****

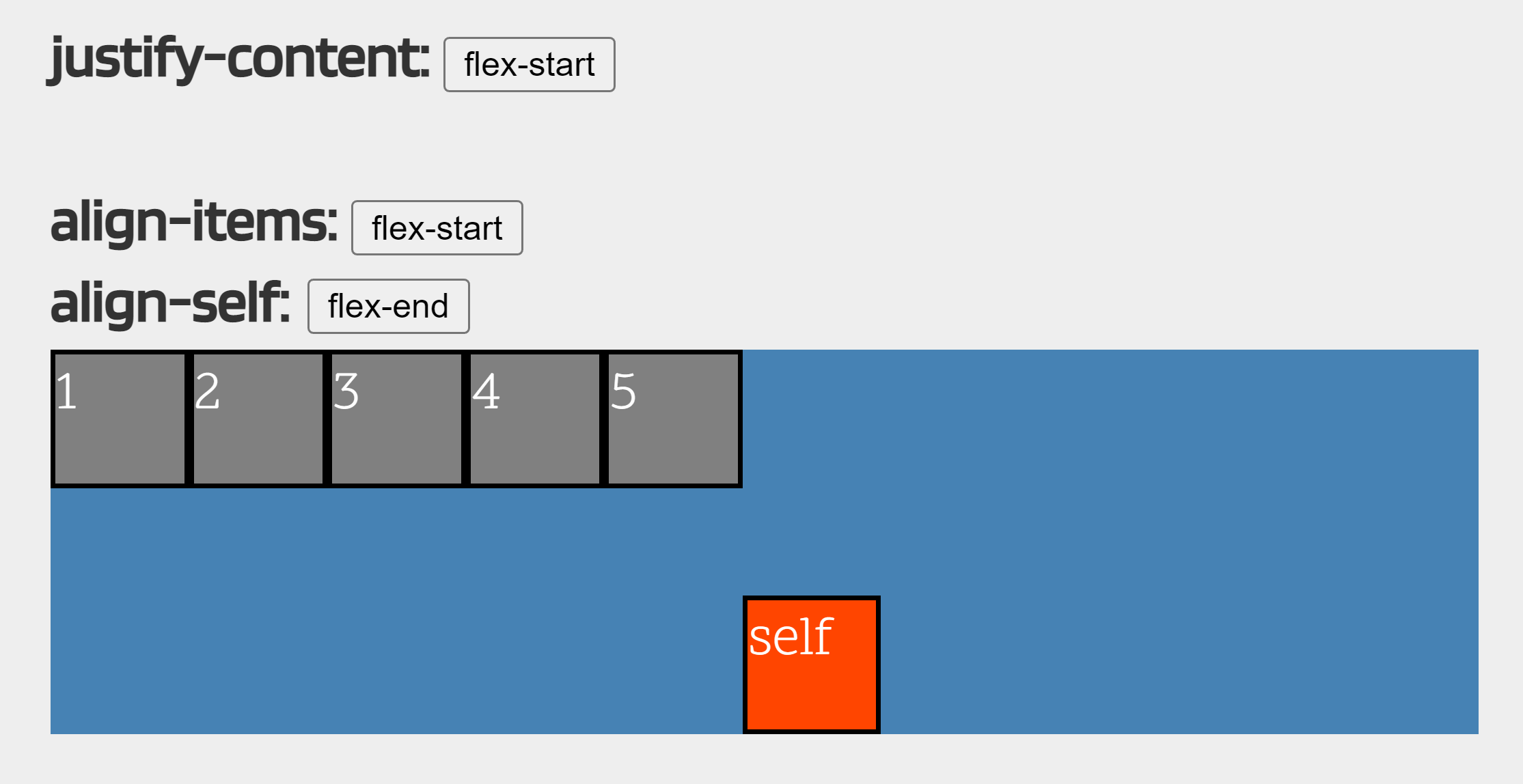
Por otro lado, la propiedad align-self actúa exactamente igual que align-items, sin embargo, es la primera propiedad de flexbox que vemos que se utiliza sobre un ítem hijo específico y no sobre el elemento contenedor. Salvo por este detalle, funciona exactamente igual que align-items.

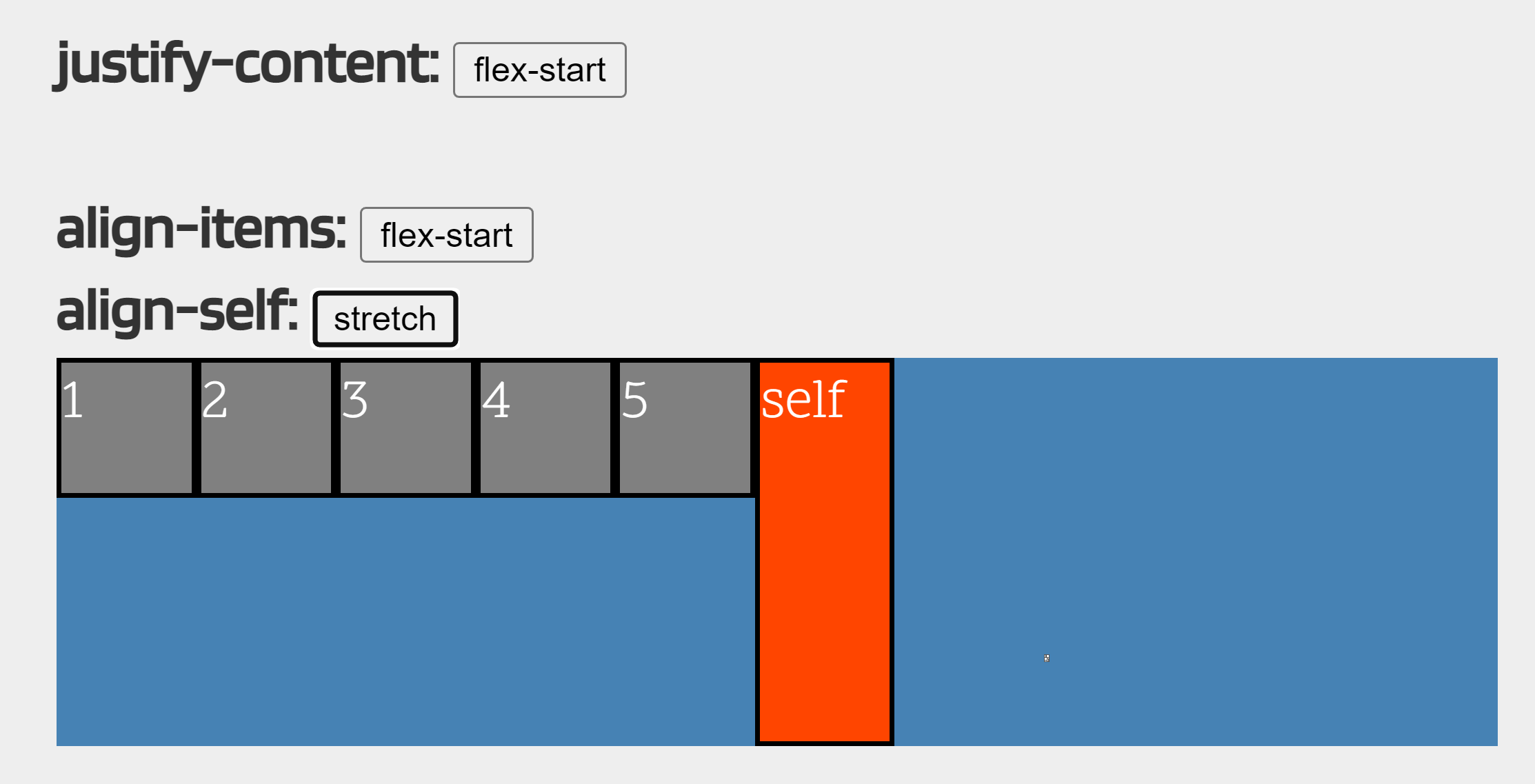
Gracias a ese detalle, align-self nos permite cambiar el comportamiento de align-items y sobrescribirlo con comportamientos específicos para ítems concretos que no queremos que se comporten igual que el resto. La propiedad puede tomar los siguientes valores:

|  |  |
| --- | --- |
| Valor | Descripción |
| flex-start | Alinea los ítems al principio del contenedor. |
| flex-end | Alinea los ítems al final del contenedor. |
| center | Alinea los ítems al centro del contenedor. |
| stretch | Alinea los ítems estirándolos al tamaño del contenedor. |
| baseline | Alinea los ítems en el contenedor según la base de los ítems. |
| ****auto**** | Hereda el valor de **align-items** del padre (si no se ha definido, es **stretch**). |

Si se especifica el valor auto a la propiedad align-self, el navegador le asigna el valor de la propiedad align-items del contenedor padre, y en caso de no existir, el valor por defecto: stretch. Veamos un par de ejemplos para verlo en funcionamiento:



****



**Atajo: Alineaciones**

Existe una propiedad de atajo con la que se pueden establecer los valores de align-content y de justify-content de una sola vez, denominada place-content:

    .container {

        display: flex;

        place-content: flex-start flex-end;

        /\* Equivalente a... \*/

        align-content: flex-start;

        justify-content: flex-end;

    }

**Propiedades de hijos**

A excepción de la propiedad align-self, todas las propiedades que hemos visto hasta ahora se aplican sobre el elemento contenedor. Las siguientes propiedades, sin embargo, se aplican sobre los ítems hijos.

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Descripción |
| flex-grow | **0** | **number** | Número que indica el factor de crecimiento del ítem respecto al resto. |
| flex-shrink | **1** | **number** | Número que indica el factor de decrecimiento del ítem respecto al resto. |
| flex-basis | **Size | content** | Tamaño base de los ítems antes de aplicar variación. |
| order | **0** | **number** | Número (peso) que indica el orden de aparición de los ítems. |

En primer lugar, tenemos la propiedad flex-grow para indicar el factor de crecimiento de los ítems en el caso de que no tengan un ancho específico. Por ejemplo, si con flex-grow indicamos un valor de 1 a todos sus ítems, tendrían el mismo tamaño cada uno de ellos. Pero si colocamos un valor de 1 a todos los elementos, salvo a uno de ellos, que le indicamos 2, ese ítem será más grande que los anteriores. Los ítems a los que no se le especifique ningún valor, tendrán por defecto valor de 0.

En segundo lugar, tenemos la propiedad flex-shrink que es la opuesta a flex-grow. Mientras que la anterior indica un factor de crecimiento, flex-shrink hace justo lo contrario, aplica un factor de decrecimiento. De esta forma, los ítems que tengan un valor numérico más grande, serán más pequeños, mientras que los que tengan un valor numérico más pequeño serán más grandes, justo al contrario de cómo funciona la propiedad flex-grow.

Por último, tenemos la propiedad flex-basis, que define el tamaño por defecto (de base) que tendrán los ítems antes de aplicarle la distribución de espacio. Generalmente, se aplica un tamaño (unidades, porcentajes, etc...), pero también se puede aplicar la palabra clave content que ajusta automáticamente el tamaño al contenido del ítem, que es su valor por defecto.

**Atajo: Propiedades de hijos**

Existe una propiedad llamada flex que sirve de atajo para estas tres propiedades de los ítems hijos. Funciona de la siguiente forma:

    .item {

        /\* flex: <flex-grow> <flex-shrink> <flex-basis> \*/

        flex: 1 3 35%;

    }

**Huecos (gaps)**

Existen dos propiedades de flexbox que han surgido recientemente: row-gap y column-gap. Dichas propiedades, permiten establecer el tamaño de un «hueco» entre ítems desde el elemento padre contenedor, y sin necesidad de estar utilizando padding o margin en los elementos hijos.

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Descripción |
| row-gap | **normal** | **size** | Espacio entre filas (sólo si flex-direction: column) |
| column-gap | **normal** | **size** | Espacio entre columnas (sólo si flex-direction: row) |

Hay que tener en cuenta que sólo una de las dos propiedades tendrá efecto, dependiendo de si la propiedad flex-direction está establecida en column o en row. Eso sí, es posible usar ambas si tenemos la propiedad flex-wrap definida a wrap y, por lo tanto, disponemos de multicolumnas flexbox.

**Atajo: Huecos**

En el caso de que queramos utilizar una propiedad de atajo para los huecos, podemos utilizar la propiedad gap:

|  |  |  |
| --- | --- | --- |
| Propiedad | Valor | Descripción |
| gap | **0** | **size** | Aplica el tamaño indicado para el hueco en ambos ejes. |
| gap | **0 0** |  **size size** | Aplica los tamaños indicados para el hueco del eje X y el eje Y. |

Y veámosla en práctica:

    .container {

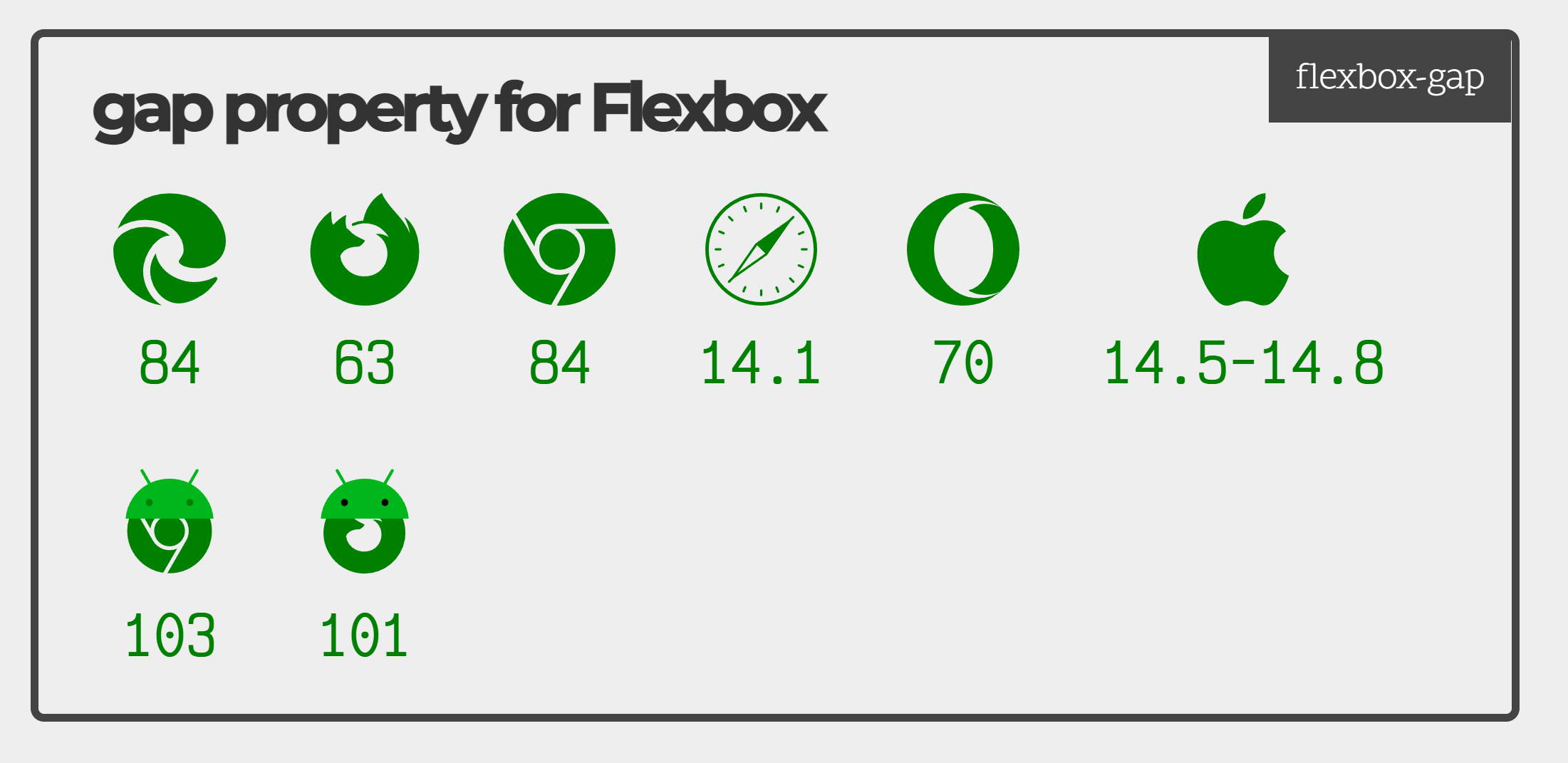
        /\* gap: <row> <column> \*/

        gap: 4px 8px;

        /\* 1 parámetro: usa el mismo para ambos \*/

        gap: 4px;

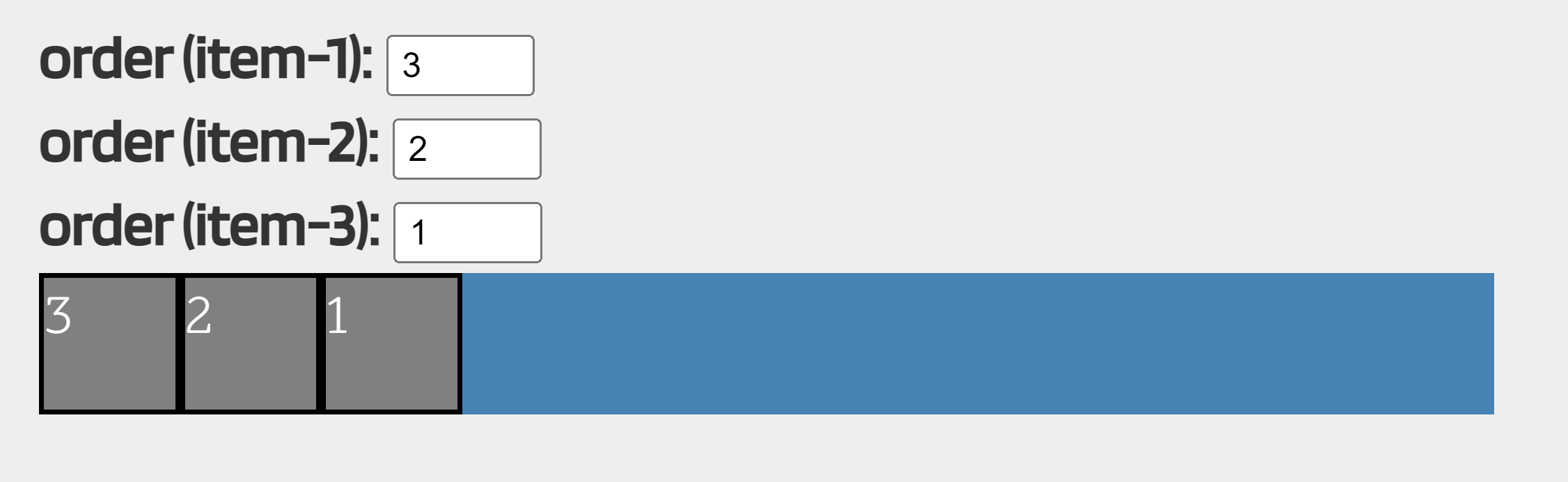
    }



**Orden de los ítems**

Por último, y quizás una de las propiedades más interesantes, es order, que modifica y establece el orden de los ítems según una secuencia numérica.

Por defecto, todos los ítems flex tienen un order: 0 implícito, aunque no se especifique. Si indicamos un order con un valor numérico, irá recolocando los ítems según su número, colocando antes los ítems con número más pequeño (incluso valores negativos) y después los ítems con números más altos.



De esta forma podemos recolocar fácilmente los ítems incluso utilizando media queries o responsive design.

