

# Diseño responsivo

Formación Profesional DAW

**Diego Martín**

*diego.martin@cenecmalaga.es*



# Índice

---

1. Diseño responsivo y breve historia
2. CSS Media queries
3. Viewport
4. Reset CSS
5. Flexbox y CSS Grid
6. Bootstrap

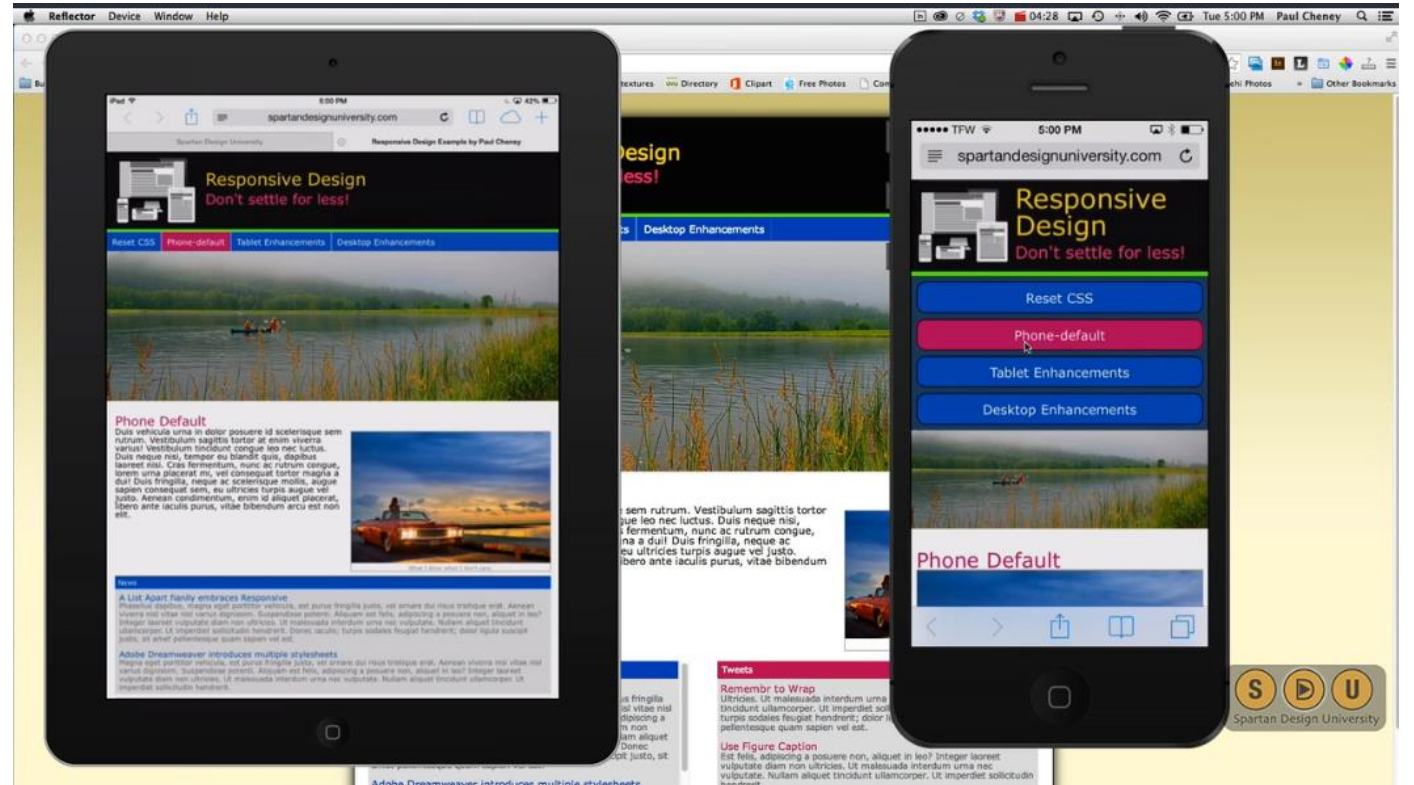
# Índice

---

1. Diseño responsivo y breve historia
2. CSS Media queries
3. Viewport
4. Flexbox
5. Grid
6. Bootstrap

# ¿Qué es el diseño responsivo?

- Una misma aplicación web es capaz de verse de diferentes formas dependiendo (i.e: resolución) del tamaño del dispositivo
- Algunas ventajas de diseño responsivo
  - Visualización óptima para cualquier dispositivo
  - Mismo contenido, misma url
  - Mayor alcance de la web
  - Solución rápida, buena y barata
  - Solución escalable



# Un poco de historia

---

- Antiguamente se utilizaba el atributo `media` de `<link>` para especificar diferentes estilos para diferentes tipos de media (e.g: all, print, screen, speech, handheld, etc.)

```
<head>
```

```
  <link rel="stylesheet" type="text/css" href="theme.css">
```

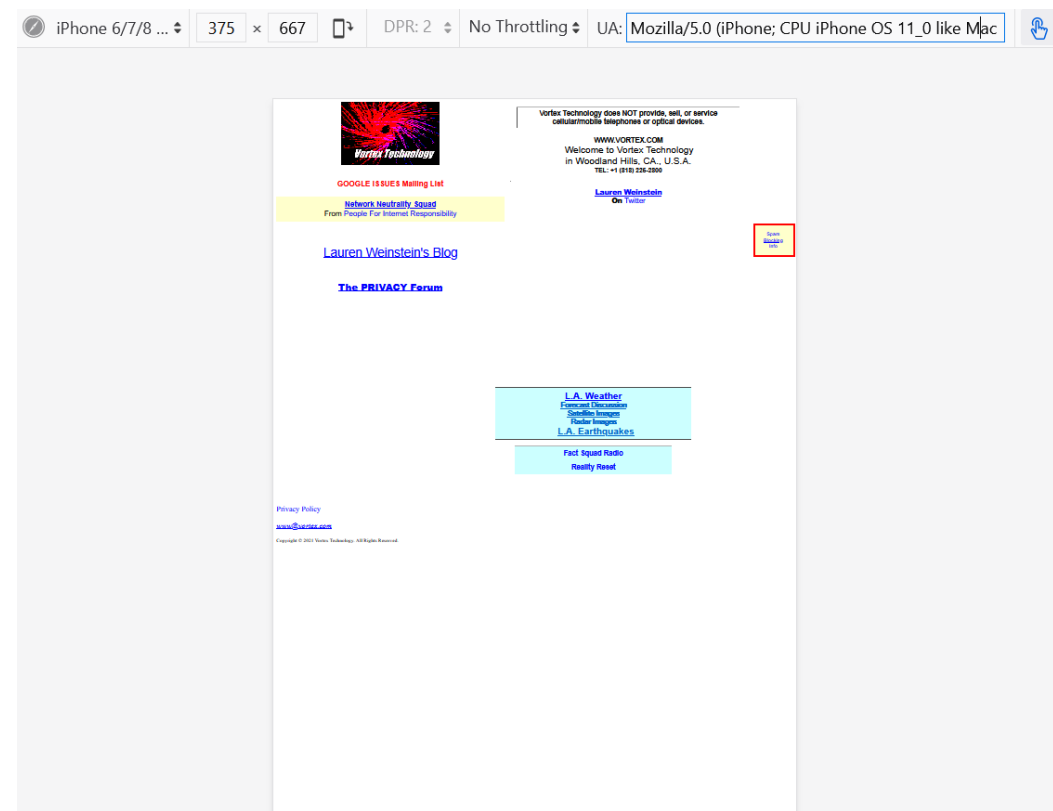
```
  <link rel="stylesheet" type="text/css" href="print.css" media="print">
```

```
</head>
```

- Tenía algunos problemas
  - Los productores de teléfonos lo ignoraban
  - Los atributos eran insuficientes para especificar inequívocamente el tipo de dispositivo o tamaño

# Comportamiento por defecto

- Ver ejemplo de una web antigua <https://www.vortex.com>
- ¿Qué hace un teléfono móvil para visualizar una web muy ancha?
- ¿Por qué se comportan así?
  - La página se escala para encajar en el dispositivo porque no sabe qué otra cosa hacer.
  - No se puede leer bien el texto porque es demasiado pequeño o hay que hacer *scroll* horizontal
- ¿Cómo solucionarlo?
  - Ignorar el problema (e.g: algunos siguen usando tablas en html)
  - Crear una aplicación móvil para mostrar el contenido
  - Crear otra web para móviles (e.g: <https://miweb.com/mobile>)
  - **CSS Media queries**



# CSS Media queries

---

- Las *media query* son una sintaxis especial de CSS que permiten que algunas reglas CSS solo se apliquen si las condiciones definidas se cumplen

```
@media not|only mediatype and|not|only(media feature)
{
  .my - code { ... }
}
```

- El uso más habitual es definir los límites en pixels de anchura mínimos o máximos a partir de los cuales se aplican las reglas CSS
- Hay dos **estrategias** responsivas:
  1. *Mobile-first* → Se crea el CSS principal para tamaño de dispositivo móvil y se usan media queries min-width para ajustar a medida que la anchura del navegador va aumentando
  2. *Desktop-first* → Se crea el CSS principal para tamaño de escritorio y se usan media queries max-width para ajustar a medida que la anchura del navegador va disminuyendo





Los ejemplos para estas sesiones utilizarán la estrategia *mobile-first*.

A modo de ejemplo, la librería Bootstrap ha utilizado *desktop-first* hasta la versión Bootstrap 3. Ahora optan por *mobile-first*.

# CSS Media queries

---

- Algunos diseñadores prefieren el *desktop-first* porque el tamaño móvil, a veces, incluye menos contenido (imágenes, iconos, etc.) que el tamaño escritorio. Es una cuestión de preferencias y de creencias
- La preferencia de *desktop-first* está motivada en gran medida por un mito por el cual se considera que el usuario móvil desea contenido más resumido porque busca movilidad y rapidez de acceso a información. La evidencia sugiere que la mayoría de usuarios utiliza el móvil incluso cuando está en casa mientras ve la tele, por lo tanto es arriesgado pensar que este tipo de usuario no se desee ver todo el contenido

Property	Desktop-First	Mobile-First
Resolution Transition	 This approach follows "Graceful Degradation" from higher resolution formats to lower resolutions	 This approach follows "Progressive Enhancement" from lower resolution to higher resolutions
Text Quantity	More	Less
Font Size	Smaller	Larger
Download Speed	Slower	Faster
Call-to-action	More than one call-to-action per page	One call-to-action per page
Access to hardware	Not so easy to access other hardware on the device	Easy access to mobile camera, microphone, flash light etc.
Best works on	Desktops	Mobile phones



# CSS Media queries con *device width*

---

- Ejemplo utilizando la anchura del dispositivo, independientemente del tamaño del navegador. El

```
@media only screen and (max-device-width: 900px){  
  /* CSS para tamaño de device hasta 900px */  
}
```

```
@media only screen and (min-device-width:901px){  
  /* CSS para tamaño de device a partir de 901px */  
}
```



La utilización de device width puede ser problemática porque ignora el tamaño del navegador

# Ejercicio 1

---

- Crea una página web simple en la que el color de fondo del body se vea de un color para dispositivos pequeños de hasta 900px y de otro color para dispositivos a partir de 901px
- Abre la página en un navegador (e.g: Firefox, Edge o Chrome) y observa cómo al modificar la anchura del navegador el estilo no cambia porque está ligado a la anchura del dispositivo

# CSS Media queries con *browser width*

- Ejemplo para diseño *mobile-first* con 3 grupos: hasta 768px, otra entre 768px y 992px y otra para anchuras de más de 992px

```
/* CSS para tamaño pequeño aquí */
```

```
@media(min-width:768px) {
```

```
  /* CSS para tamaño entre 768px y 992px */
```

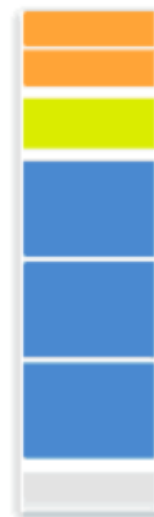
```
}
```

```
@media(min-width:992px) {
```

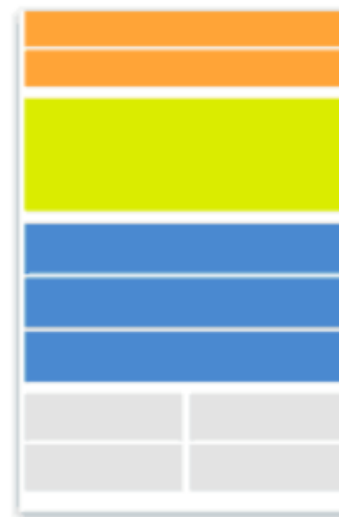
```
  /* CSS para tamaño a partir de 992px */
```

```
}
```

← hasta 768px →



← hasta 992px →



← a partir de 992 px →



La utilización de *browser width* es preferible por permitir un diseño más robusto que el *device width*.

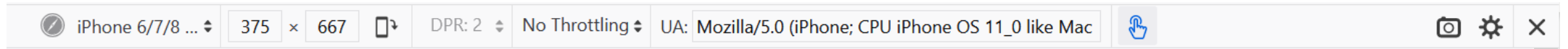
La cantidad de formas diferentes de visualización es opcional y debe aplicar el sentido común.

Los límites de las media queries también son opcionales y se pueden usar los que se consideren oportunos.

# Ejercicio 2

---

- Crea una página web simple en la que el color de fondo del body se vea de un color para tamaños de pantalla pequeña de hasta 768px, de otro color para tamaños de hasta 992px y de otro color para tamaños anchos
- Abre un navegador (e.g: Firefox, Edge o Chrome) y utiliza las herramientas de desarrollador para visualizar cómo cambia el diseño cuando se alcanzan los tamaños especificados
- En las herramientas del desarrollador, utiliza algún dispositivo como el iPhone7 y observa cómo a pesar de que la anchura sea de 375px, el media query aplicado es el de tamaños entre 768px y 992px. ¿Por qué?



- Modifica el ejemplo para mostrar diferentes imagenes de fondo en lugar de colores, y un texto *Hola mundo* y observa el mismo efecto al utilizar dispositivos específicos como el iPhone 7 en el que no parece aplicarse el media query de 375px. Observa también cómo el texto se encoje

# Viewport o ventana virtual

---

- El **viewport** del navegador, también llamado ventana virtual, es el area de la ventana en la que el contenido se puede ver
  - no suele coincidir con el tamaño de la web renderizada, por lo que el navegador provee de *scrollbars*
  - los dispositivos muy estrechos renderizan páginas en un viewport normalmente más ancho que la ventana, haciendo que el contenido se encoja para poderse ver todo a la vez (e.g: ver <https://www.vortex.com> con un dispositivo móvil como iPhone7)
  - Por ejemplo, si una pantalla de móvil tiene un *width* de 640px, las páginas podrían renderizarse con un viewport de 980px cuyo contenido se encojería para entrar dentro de los 640px
  - Esto se hace así porque muchas páginas no están optimizadas para dispositivos móviles y el contenido en éstos se ve mal. Con el *viewport* se consigue que estas web no optimizadas para móvil se vean mejor en estas pantallas estrechas



Los dispositivos **mienten** acerca de su verdadero tamaño, pero esto se puede remediar si queremos un control total sobre el diseño

## Ejercicio 3

---

- Haz que en el ejercicio anterior, todos los dispositivos (e.g: iPhone 7), dejen de mentir y utilicen el diseño acorde a los pixels de su pantalla. En otras palabras, utiliza la *meta tag viewport*
- Observa cómo el tipo de dispositivo ya no afecta al diseño, y que éste es fiel al tamaño real

# Diferencias en estilos CSS de cada *user agent*

---

- Cada navegador tiene una serie de estilos predefinidos para algunas etiquetas y éstos pueden diferir → **problema de inconsistencia**
- Por ejemplo, las etiquetas <h1> hasta <h6> en cada navegador se ven de forma diferente. Por lo general el tamaño de sus fuentes aumenta, se visualizan en negrita y añaden márgenes por arriba y por abajo
- Para resolver estos problemas existen 2 estrategias
  1. Normalizar CSS → Es una solución más suave consistente en compensar las diferencias.  
Ver <https://github.com/necolas/normalize.css/blob/master/normalize.css>
  2. Resetear CSS → Es una solución más agresiva consistente en eliminar completamente los estilos por defecto que los navegadores incluyen, reseteando todos los estilos que vienen en el *user agent*  
Ver <https://meyerweb.com/eric/tools/css/reset>



Los partidarios de la solución de normalizar CSS argumentan que el reseteo es "feo" porque incluye muchos selectors y complica el depurado

# Diferencias en estilos CSS de cada *user agent*

---

```
html, body, div, span, applet, object, iframe, h1, h2, h3, h4, h5, h6, p, blockquote, pre, a, abbr, acronym, address, big, cite, code, del, dfn, em, img, ins, kbd, q, s, samp, small, strike, strong, sub, sup, tt, var, b, u, i, center,
dl, dt, dd, ol, ul, li, fieldset, form, label, legend, table, caption, tbody, tfoot, thead, tr, th, td, article, aside, canvas, details, embed, figure, figcaption, footer, header, hgroup, menu, nav, output, ruby, section, summary,
time, mark, audio, video {
    margin: 0;
    padding: 0;
    border: 0;
    font-size: 100%;
    font: inherit;
    vertical-align: baseline;
}
/* HTML5 display-role reset for older browsers */
article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
    display: block;
}
body {
    line-height: 1;
}
ol, ul {
    list-style: none;
}
blockquote, q {
    quotes: none;
}
blockquote:before, blockquote:after,
q:before, q:after {
    content: '';
    content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
```



Hay muchos ejemplos en la web de reseteo de CSS. Este es uno que se obtiene de <https://meyerweb.com/eric/tools/css/reset>



# Ejercicio 4

---

- Crea una web solamente con HTML sin CSS con varias líneas de texto `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`
- Abre esta web con diferentes navegadores y observa las diferencias y los estilos que cada navegador aplica
- Incluye reseteo CSS y vuelve a observar cómo se renderiza el texto

# Flexbox – qué es y para qué sirve

---

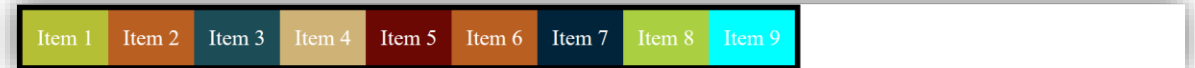
- **Flexbox** es una propiedad de CSS3 que permite que los elementos se comporten de forma predecible en diseño responsivo
- Antiguamente se hacía un uso excesivo de la propiedad `float` de CSS, pero Flexbox reemplaza ese mecanismo arcaico
- La estructura se compone de:
  1. El *flex container* o contenedor flex
  2. Los *flex items* o elementos flex
- Utiliza dos ejes o *flexbox axes*:
  1. *Main axis* o eje principal
  2. *Cross axis* o eje secundario



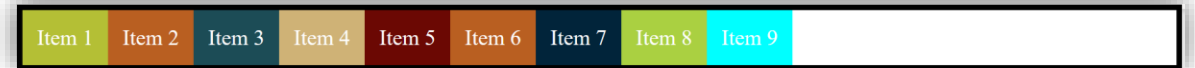
# Flexbox – propiedades

- **display** → comportamiento del contenedor en cuanto a visualización. Es necesario que exista y hay dos posibles opciones

- *inline-flex* para contenedor en línea (similar a *inline-block*)

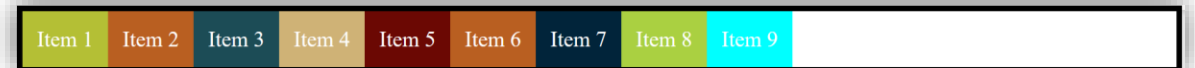


- *flex* para contenedor en bloque (similar a *block*)

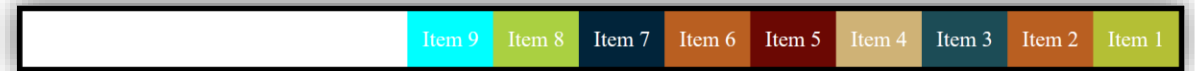


- **flex-direction** → dirección, tiene 4 valores

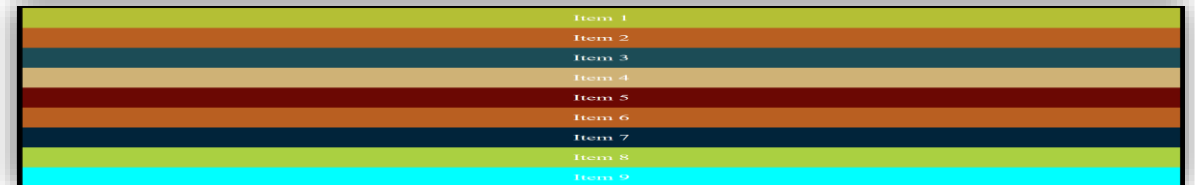
- *row* es el valor por defecto, de izquierda a derecha



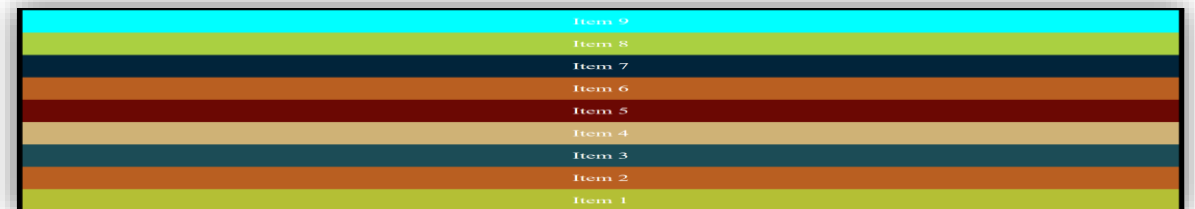
- *row-reverse* de derecha a izquierda



- *column* de arriba hacia abajo



- *column-reverse* de abajo hacia arriba



# Flexbox – propiedades

- ***flex-wrap*** → comportamiento del contenedor en cuanto a desbordamiento de sus elementos. Acepta 3 valores

- *nowrap* es el valor por defecto, una línea sin desbordamiento



- *wrap* es multilínea, con desbordamiento (el último baja)



- *wrap-reverse* es multilínea pero inverso (el ultimo sube)



**NOTA:** *flex-wrap* puede funcionar en columnas si se cambia *flex-direction* a *column*; y se establece un *height* en el contenedor (pruébalo)

- ***flex-flow*** → atajo para resumir los valores de *flex-direction* y *flex-wrap*. Ejemplo: `flex-flow: row-reverse wrap;`

- ***justify-content*** → alineación de los elementos en el eje principal

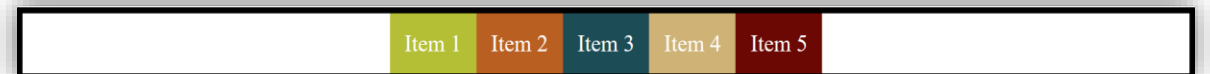
- *flex-start* es el valor por defecto, al comienzo del eje



- *flex-end* al final del eje

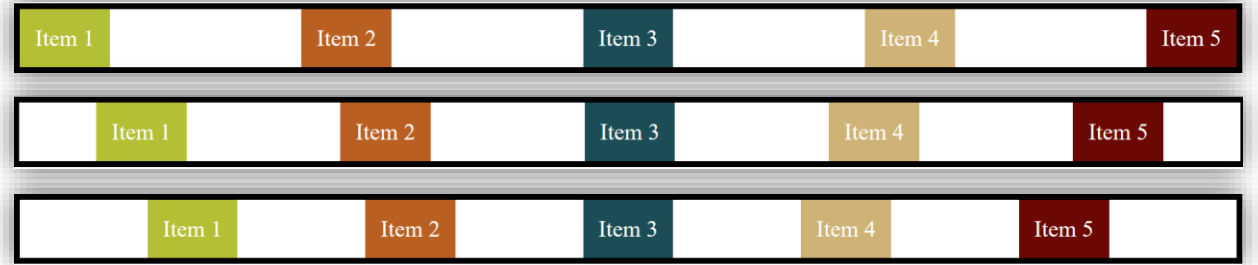


- *center* en el centro del eje



# Flexbox – propiedades

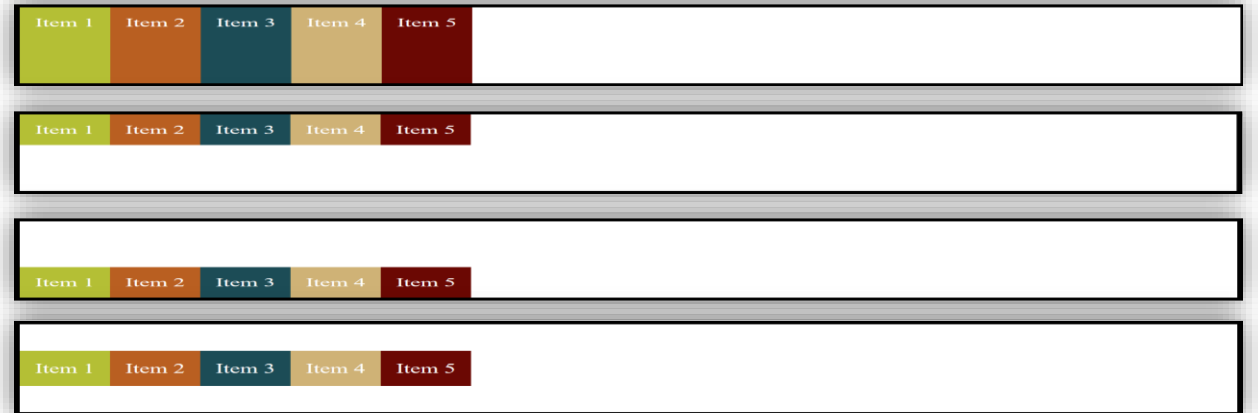
- *space-between* para espacio entre medias
- *space-around* para espacio alrededor
- *space-evenly* para espacio equivalente



**NOTA:** *justify-content* puede funcionar en columnas si se cambia *flex-direction* a *column*; y se establece un *height* en el contenedor (pruébalo)

- ***align-items*** → alineación de los elementos en el eje secundario

- *stretch* es el valor por defecto
- *flex-start* los agrupa al comienzo del eje
- *flex-end* los agrupa al final del eje
- *center* los agrupa en el centro



**NOTA:** hay otros valor como *baseline* que permite alinearlos según la base del contenido de los ítems en el contenedor (prueba a dar diferentes padding a los ítems)

# Flexbox – propiedades

---

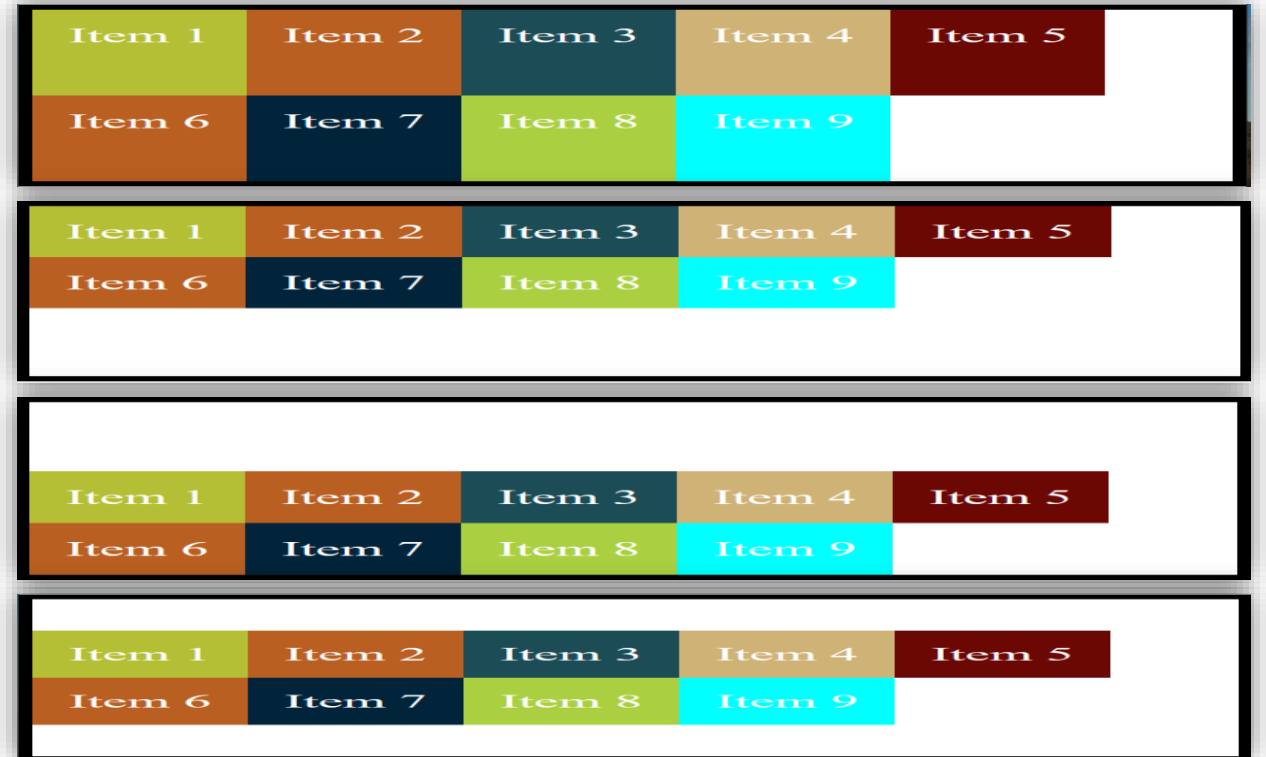
- ***align-content*** → alineación de cada una de las líneas del contenedor multilinea

- *stretch* es el valor por defecto

- *flex-start* los agrupa al comienzo del eje principal

- *flex-end* los agrupa al final del eje principal

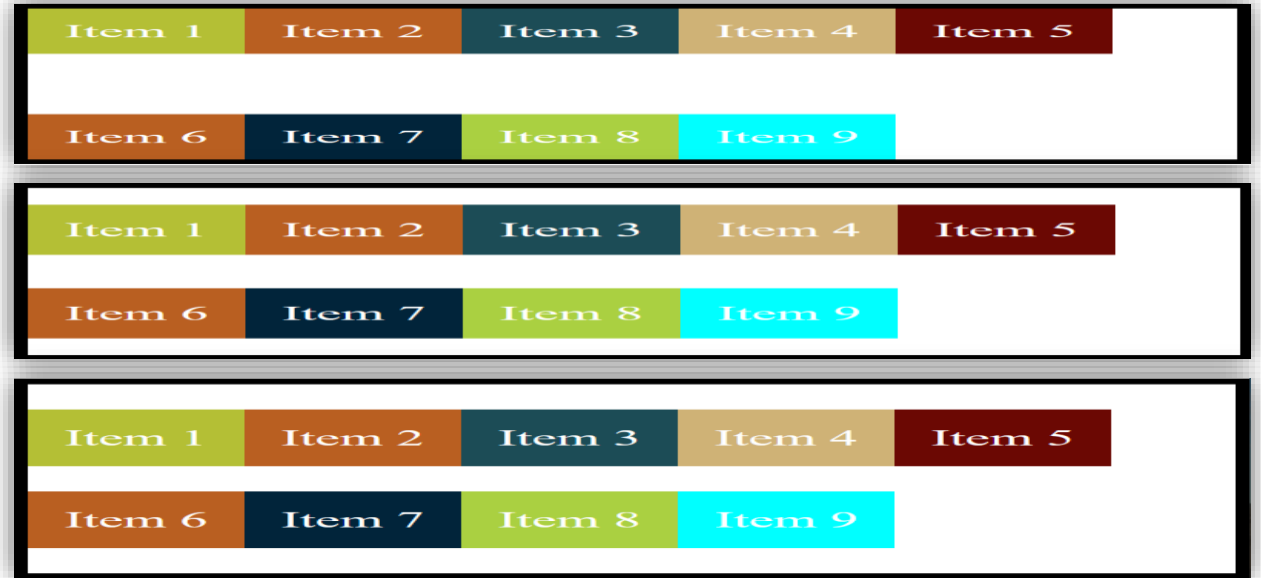
- *center* los agrupa en el centro



# Flexbox – propiedades

---

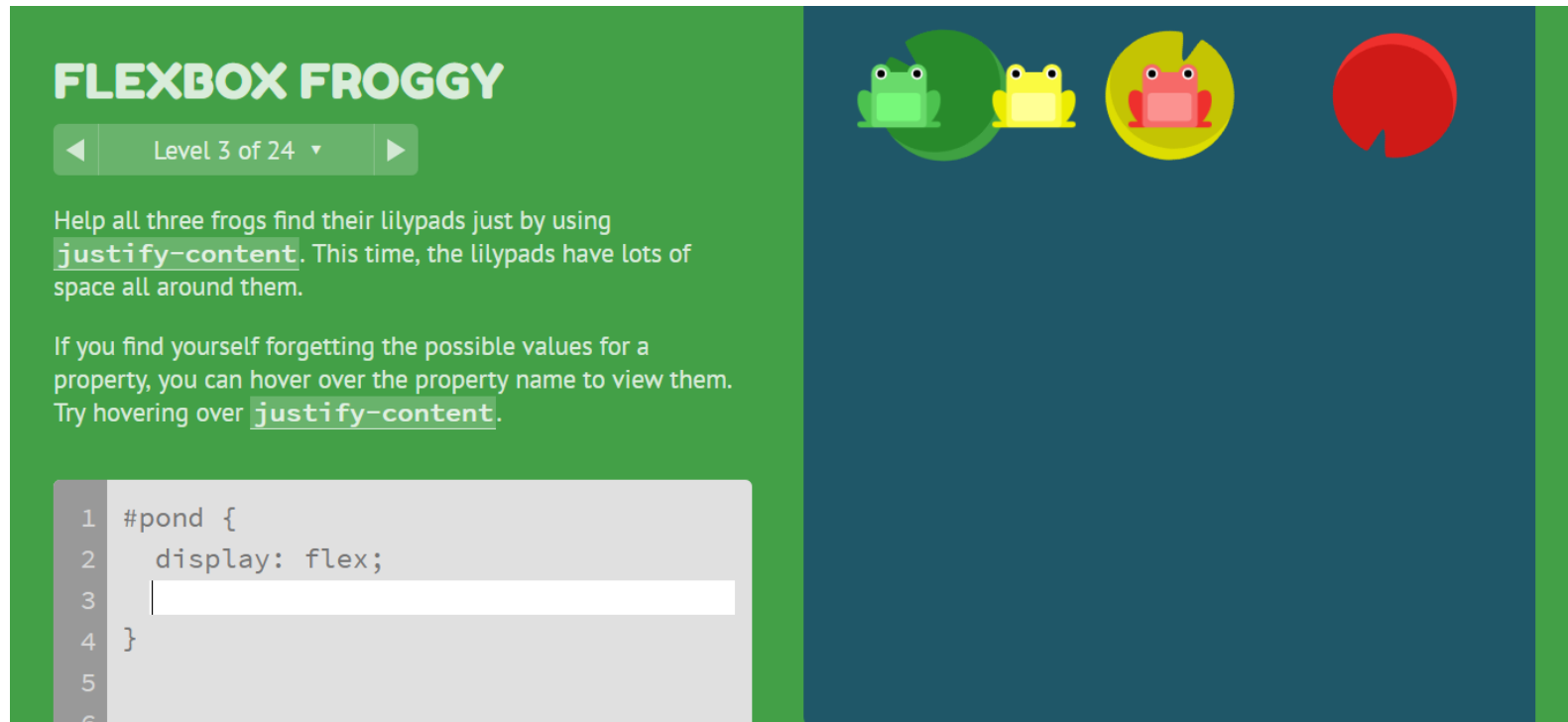
- *space-between* para espacio entre las líneas
- *space-around* para espacio alrededor de las líneas
- *space-evenly* para espacio equivalente



# Ejercicio 5 – Aprende Flexbox jugando

---

- Accede a <https://flexboxfroggy.com> y realiza los 24 niveles en los que pondrás en práctica conocimientos de flexbos adquiridos
- Tienes que ayudar a que las ranas se posen en el nenúfar





# Referencias

---

- <https://lenguajecss.com/css/maquetacion-y-colocacion/flexbox>