

As easy as pie chart

Intro

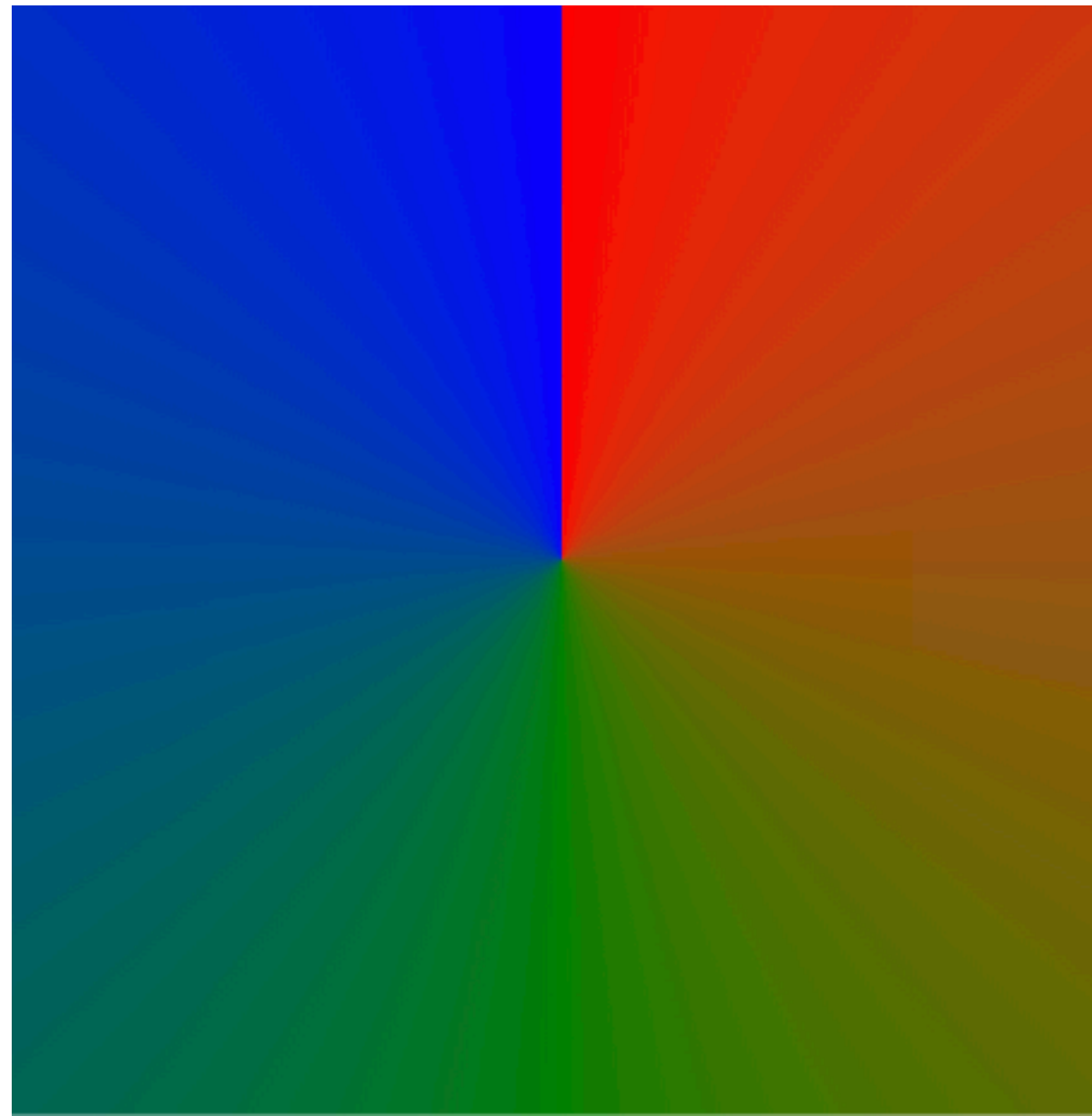
VADYM HORBONOS



WOMAN IN RED PIN-UP BY
STUART IMMOMEN

Conic gradient

— це вид градієнту при якому колор стопи розміщуються по окружності кола



Conic gradient — Draft

— не увійшов до специфікації CSS 3. Зараз знаходиться у драфті CSS image module 4

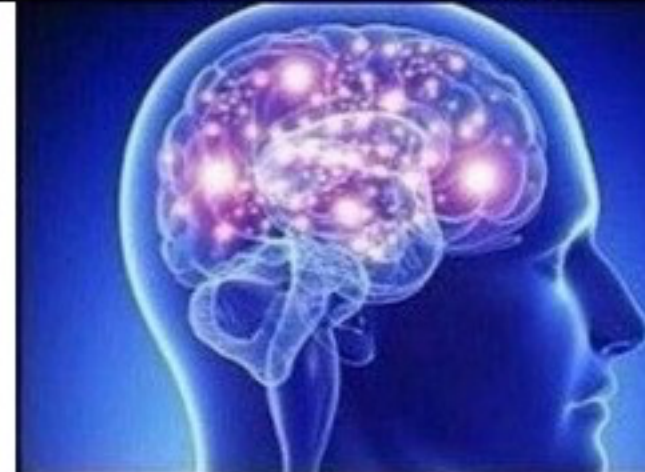
<https://www.w3.org/TR/css-images-4/#conic-gradients>

Conic gradient Evolution

— це
градієнт



— це
конусоподібний
градієнт



— градієнт, що
схожий на radial,
але трошки інший



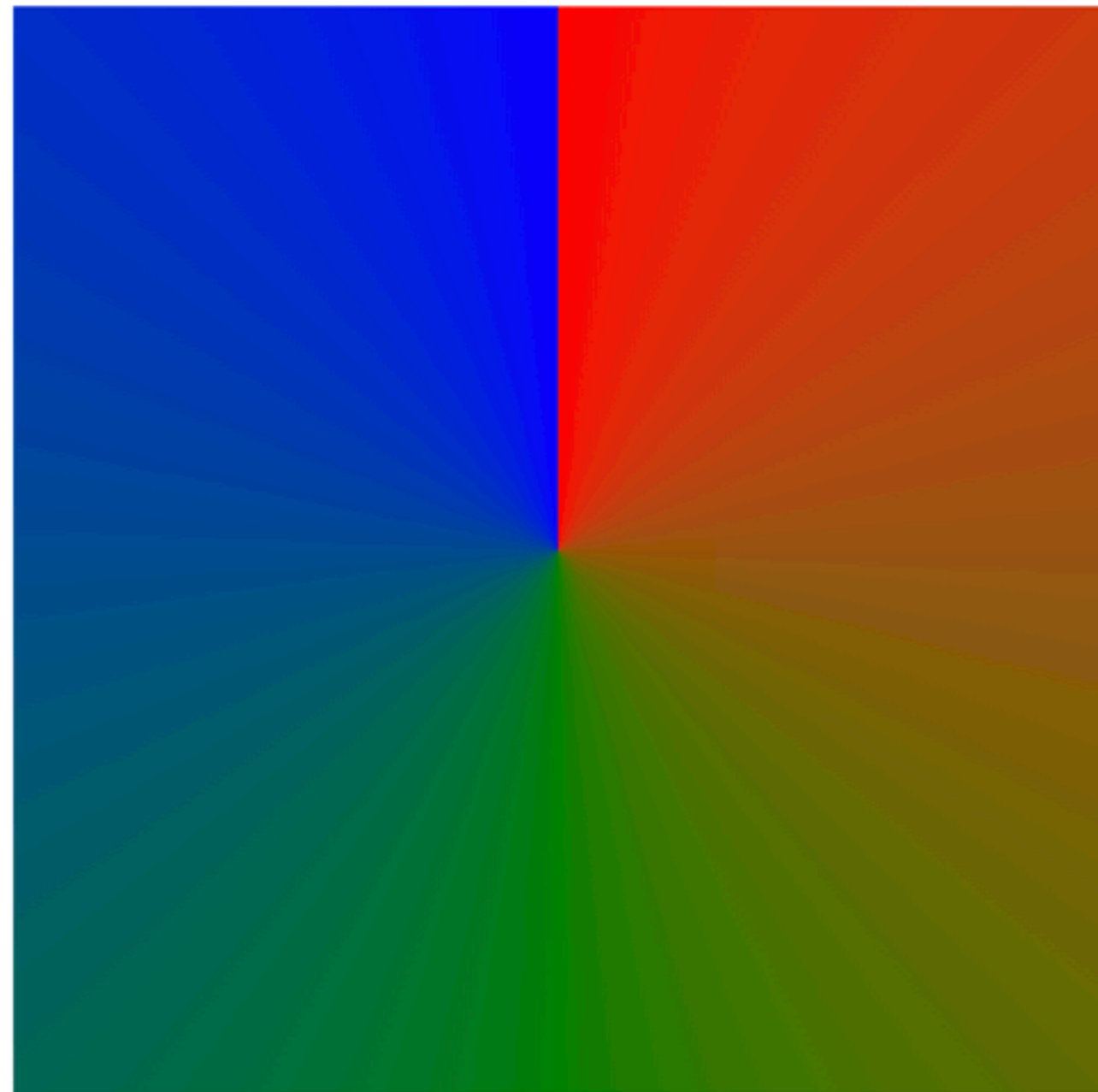
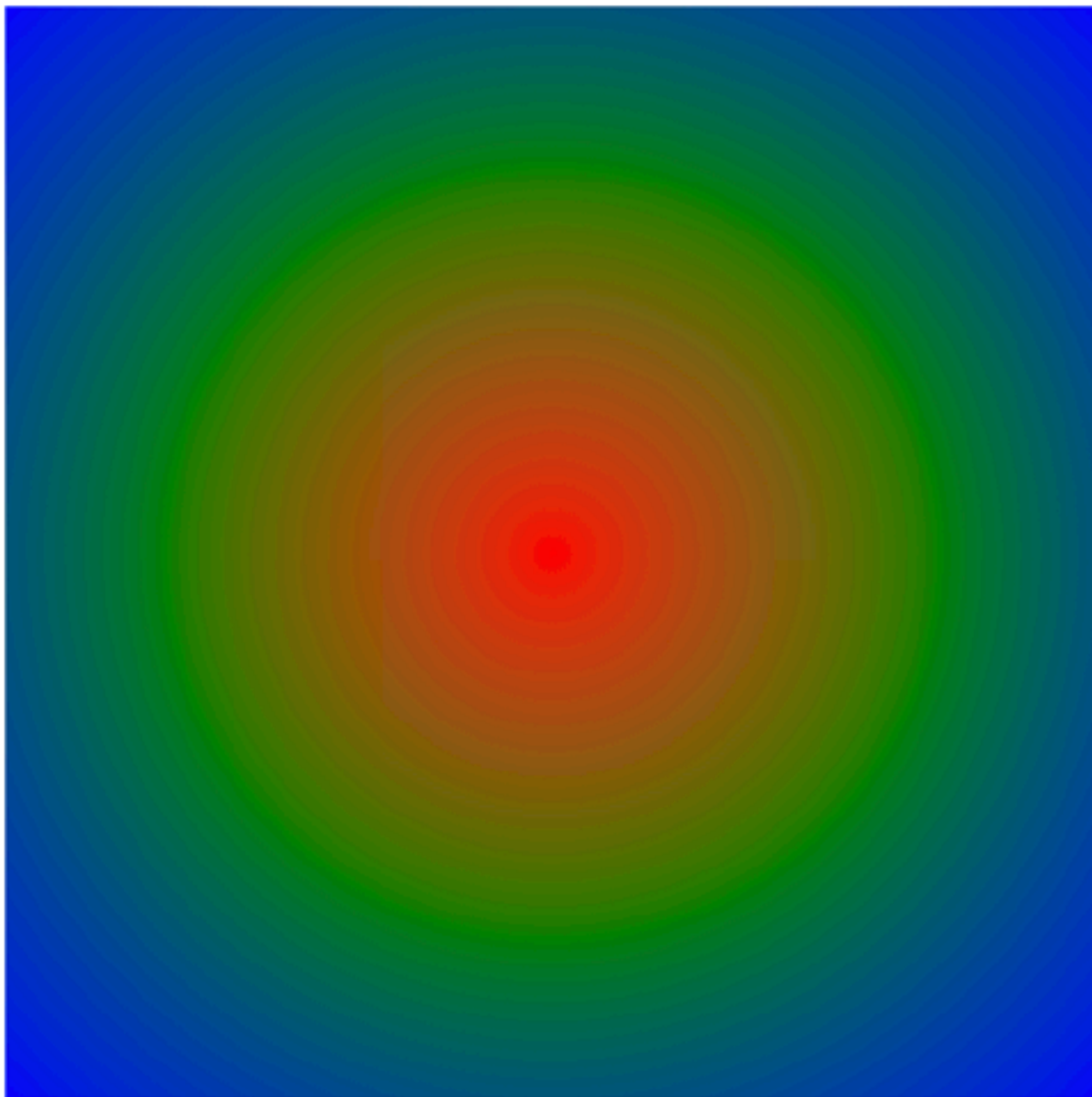
— це вид градієнту
при якому колор
стопи розміщуються
по окружності кола



Conic vs Radial

Comparison

<https://codepen.io/vadym1930/pen/VwYNWKQ>



Browser support

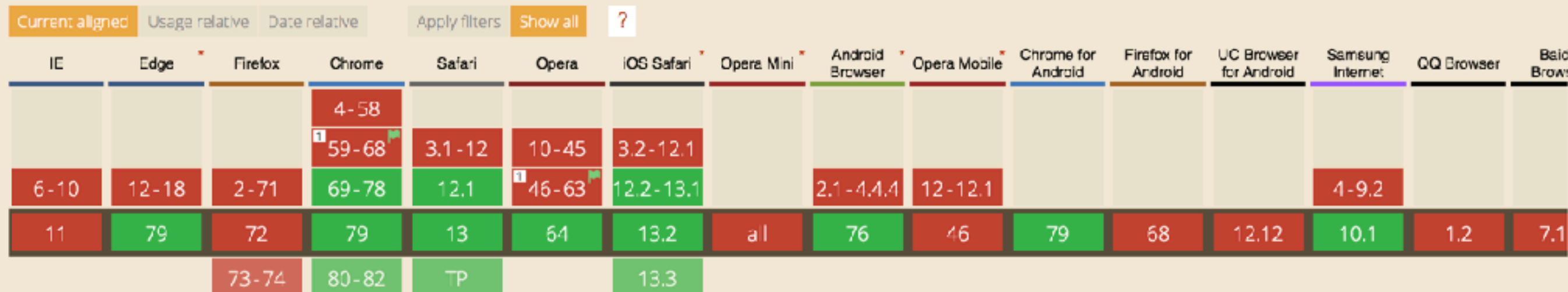
<https://caniuse.com/#search=conic>

<https://leaverou.github.io/conic-gradient/> — polyfill



Usage	% of	all users	?
Global			79.88%

Method of defining a conical or repeating conical color gradient as a CSS image.



Dynamic Pie chart

changing the CSS properties for the DOM element

Approaches

- 1) INLINE STYLES goes from template
- 2) INLINE DATA-ATTR goes from template to manipulate with JS
- 3) CK EDITOR PLUGIN
- 4) Grab data via JSON API
- 5) Grab data via GRAPHQL
- 6) DO NOT USE, go with one of the contributed modules

Approaches Why

1) INLINE STYLES goes from template

— can't apply polyfill without preprocess, not ok

2) INLINE DATA-ATTR goes from template to manipulate with JS

— ok

3) CK EDITOR PLUGIN

— better than ok

4) Grab data via JSON API

— ok, if you do not afraid to grab
paragraph inside paragraph inside paragraph ...

5) Grab data via GraphQL

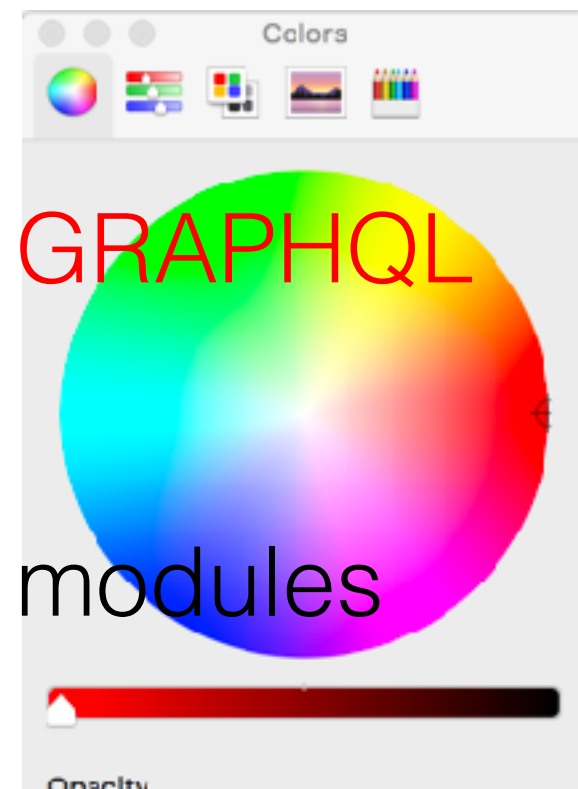
— ok

6) DO NOT USE, go with one of the contributed modules

— ok

Approaches Choice

- 1) INLINE STYLES goes from template
- 2) INLINE DATA-ATTR goes from template to manipulate with JS
- 3) CK EDITOR PLUGIN
- 4) Grab data via JSON API
- 5) Grab data via GRAPHQL
- 6) DO NOT USE, go with one of the contributed modules



Basis

```
pieStyles() {  
  let acum = 0;  
  let styles = [  
    { color: "#00A37A", value: 40 },  
    { color: "#365164", value: 30 },  
    { color: "#a54f93", value: 30 }  
  ].map(  
    segment => `${segment.color} 0 ${acum +=  
segment.value}%`  
  );  
  
  return {  
    background: `conic-gradient( ${styles.join(",")} )`  
  };  
}
```

<https://vuedose.tips/tips/the-most-modern-pie-chart-component-using-css-conic-gradient-and-vue-js/>

Helper

```
2  /**
3   * Generates comma separated string with color stops values
4   * Accept an array of objects { color: 'valid color', value: number }
5   *
6   * @param {array} arr
7   *
8   * @returns string
9   */
10 generateStyle(arr = []) {
11   if (!arr[Symbol.iterator]) {
12     arr = [];
13   }
14
15   let acum = 0;
16   let styles = arr.map(
17     segment => `${segment.color} 0 ${(acum += Number(segment.value))}%`
18   );
19
20   return styles.join(",");
21 }
```

План

1. Отримати дані через GraphQL
2. Адаптувати дані
3. Написати юніт тести
4. Переконатися, що це працює для всіх параграфів на сторінці
5. Додати поліфіл

План підготовчих дій

1. Продумати заповнення контенту та сеттінгів для пай чартів
2. Підключити необхідні бібліотеки, налаштувати зібрання бандлу

Description

Drupal Content

Пай чарт в адмін панелі може мати вигляд параграфа

body p

[About text formats](#)

[Show row weights](#)

Labels

Label cne

✶

Label two

✶

✦

4

Add another item

[Show row weights](#)

Values

55

➤

45

3

→

Add another item

[Show row weights](#)

Colors

✦

crimson

→

time

Ideal Settings look something like this

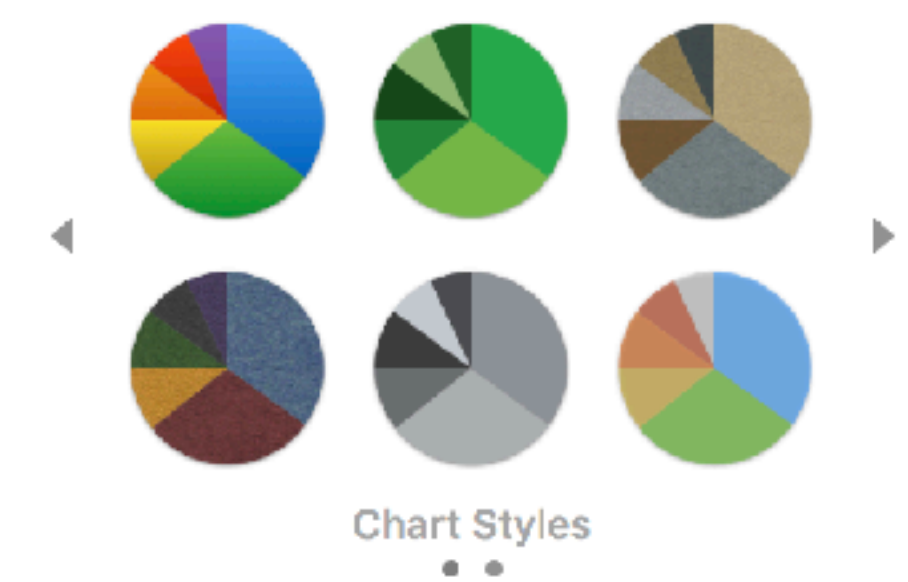


Chart Options

- ☐ Title
- ☐ Legend

Chart Font

Helvetica

Light

A

A

Chart Colors

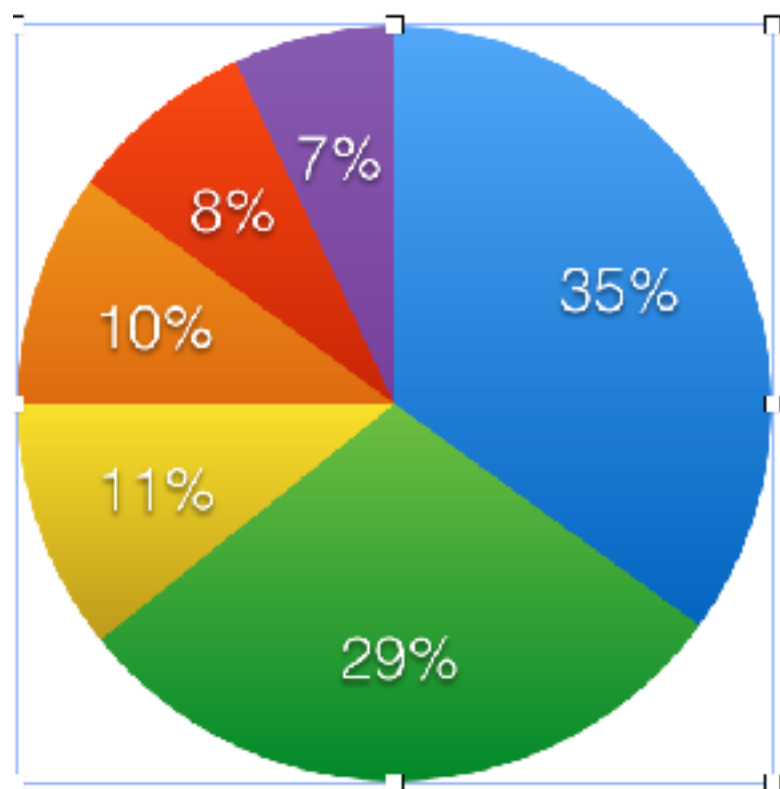


▼ Background

No Background Fill

▼ Shadow

No Shadow



CK Editor plugin
could look like this

Edit Chart Data

Chart Data							
		April	May	June	July	August	September
	Region 1	91	76	28	26	21	18

GraphQL query

GraphiQL

▶ Prettify History

```
1
2
3 query ($nid: String!) {
4   nodeQuery (filter: {conditions: [
5     {operator: EQUAL, field: "type", value: ["page"]}
6     {operator: EQUAL, field: "nid", value: [$nid]}
7   ]}) {
8     count
9     entities {
10       ...on NodePage{
11         nid
12         title
13         fieldPieChart {
14           entity{
15             ... on ParagraphPieChart {
16               fieldColors
17               fieldLabels
18               fieldDescription {
19                 processed
20               }
21               fieldValues
22             }
23           }
24         }
25       }
26     }
27   }
28 }
```

QUERY VARIABLES

```
1 {
2   "nid": "1"
3 }
```

Graphql response

```
{
  "data": {
    "nodeQuery": {
      "count": 1,
      "entities": [
        {
          "nid": 1,
          "title": "Title",
          "fieldPieChart": [
            {
              "entity": {
                "fieldColors": [
                  "crimson",
                  "lime"
                ],
                "fieldLabels": [
                  "Label one",
                  "Label two"
                ],
                "fieldDescription": {
                  "processed": "<p>Description</p>\n"
                },
                "fieldValues": [
                  "55",
                  "45"
                ]
              }
            },
            {
              "entity": {
                "fieldColors": [
                  "blue",
                  "pink"
                ],
                "fieldLabels": [
                  "Label one",
                  "Label two"
                ],
                "fieldDescription": {
                  "processed": "<p><strong>One more chart</strong></p>\n"
                },
                "fieldValues": [
                  "30",
                  "70"
                ]
              }
            }
          ]
        }
      ]
    }
  }
}
```

1

2

Two entities available

Drupal integration

Варіант гібриду

- Рутінг, безпека, контент регулюються Drupal.
- Вигляд сторінок повністю керовані JavaScript бібліотекою.
- Для клієнту гібриду підійдуть легкі, гнучкі бібліотеки. Не має особливої різниці, якщо ви виберете React або Vue.js або Svelte.

У якості бонусу ви можете отримати досить швидкий перший рендер, якщо передасте з бекенду всі дані для дефольтного рендеру сторінки

node--page.html.twig ●

docroot > themes > > templates > node--page.html.twig

```
20
21 {{ attach_library('draft_theme/js-app') }} <!-- Підключаємо бібліотеку -->
22
23 <script>
24     window.nodeIdForPieChart = "{{ node.id }}";
25 </script> <!-- Передаємо у глобальний скоуп id ноди (там міг бути весь контент
26         сторінки в json) -->
27 <div{{ content_attributes }}>
28     <div id="app"></div>
29 </div>
30
31 </article>
32 <!-- П'ємо каву -->
```

Hybrid

Libraries organisation

18

19 `js-vendors:`

20 `js:`

21 `pie-chart-app/dist/chunk-vendors.js: {}`

22

main JS vendors

23 `js-app:`

24 `css:`

25 `theme:`

26 `pie-chart-app/dist/app.css: {}` # main CSS

27 `js:`

28 `pie-chart-app/dist/app.js: {}` # main JS

29 `dependencies:`

30 `- adminimal_theme/js-vendors`

31 `- adminimal_theme/conic-polyfill-prefix`

32 `- adminimal_theme/conic-polyfill-gradient`

33

34 `conic-polyfill-prefix:`

35 `js:`

36 `pie-chart-app/shared/prefix-free.js: {}`

37

38 `conic-polyfill-gradient:`

39 `js:`

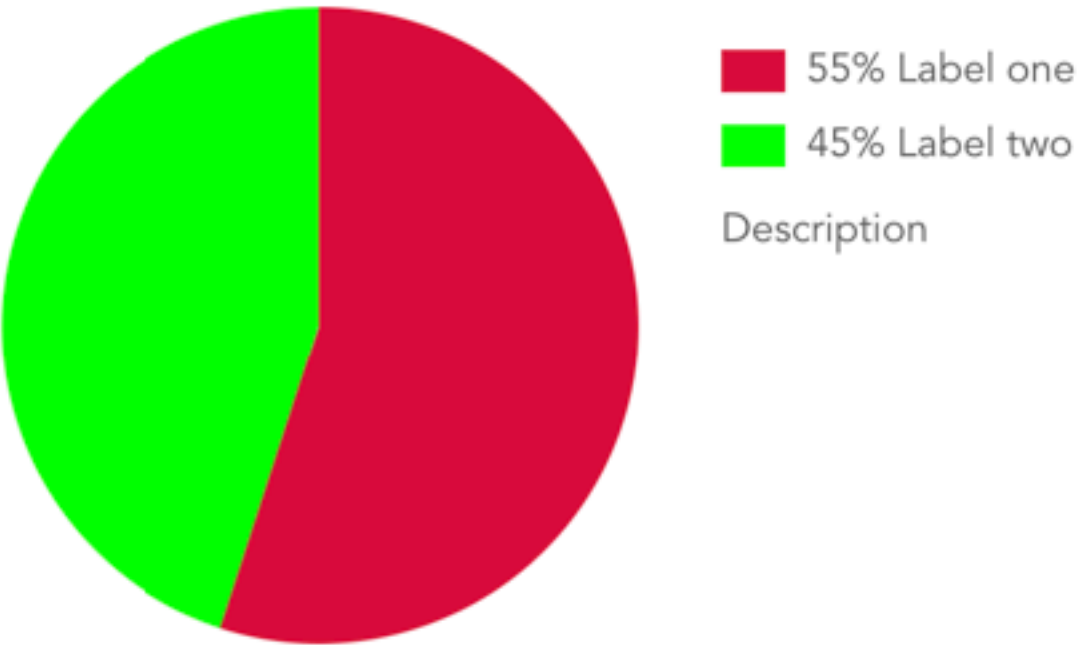
40 `pie-chart-app/shared/conic-gradient.js: {}`

41

polyfills



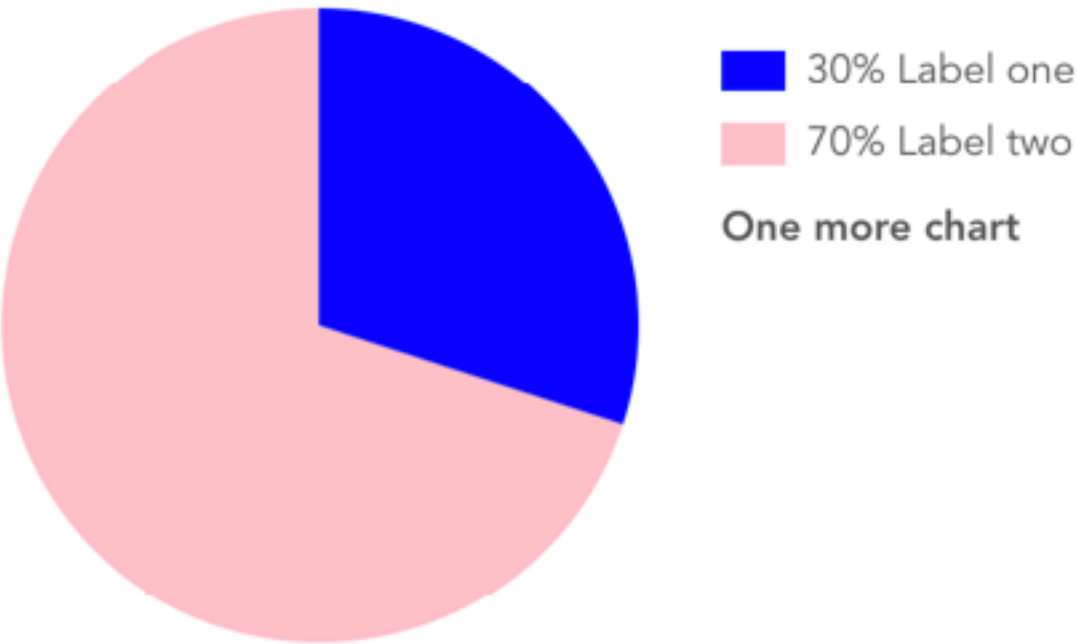
Result



55% Label one

45% Label two

Description



30% Label one

70% Label two

One more chart

Resume

— На основі простої CSS властивості можна побудувати один із найпопулярніших видів чартів

— При прийнятті рішення, яким чином реалізувати пай чарти варто враховувати

1. Наскільки кастомізованим необхідний кінцевий результат
2. Чи необхідні інші види чартів у майбутньому
3. Які ресурси ви маєте (бекенд, фронтенд)
4. Чи готові написати тести для кастомного рішення
5. Інші

— Загальне правило не використовувати кастомних рішень, крім випадків коли не має можливості застосувати готове рішення

— Не виключено, що бібліотека для пай чартів під капотом використовує подібну технологію

— Експериментуючи ми більше розуміємо як працюють речі, що дозволяє свідоміше використовувати готові рішення

Resources

- 1 <https://www.w3.org/TR/css-images-4/#conic-gradients>
- 2 <https://css-tricks.com/snippets/css/css-conic-gradient/>
- 3 <https://developer.mozilla.org/en-US/docs/Web/CSS/conic-gradient>
- 4 <https://leaverou.github.io/conic-gradient/>
- 5 Check out client app source code
<https://github.com/vadym1930/pie-chart-vue>