Nombre del Proyecto

SERVICIO DE DESARROLLO EVOLUTIVO Y ADAPTATIVO DE LA API Y SUS FUNCIONES, MÓVILES DE VISUALIZACIÓN Y SERVICIOS WEB DEL CENTRO NACIONAL DE INFORMACIÓN GEOGRÁFICA

Cliente

Centro Nacional de Información Geográfica

Fecha

15/08/2021

Versión

1.0

Tipo de documento

Informe de migración

1. Introducción

La versión 6.0 incluye 1780 commits de 544 PR desde la versión 5.3.

Una característica importante de esta versión es la capacidad de componer capas con diferentes tipos de renderizador. Anteriormente, el mapa usaba una única estrategia de representación, y todas las capas en su mapa tenían que implementar esa estrategia. Ahora es posible tener un mapa con capas que utilizan diferentes tecnologías de representación. Esto hace posible, por ejemplo, tener una capa Canvas (2D) compuesta junto con una capa basada en WebGL en el mismo mapa. También es posible crear capas con renderizadores personalizados. Por lo tanto, podría tener un mapa que use otra biblioteca (como d3) para representar una capa y usar OpenLayers para representar las otras capas. Continuaremos aprovechando esta nueva flexibilidad en futuras versiones.

Además, la versión 6.0 incluye una serie de mejoras en la representación de teselas vectoriales y debería tener una huella de memoria más baja en general. El lanzamiento también incluye una serie de características experimentales que aún no forman parte de la API estable. Eche un vistazo a los ejemplos de un nuevo renderizador basado en WebGL y la función experimental useGeographic().

Esta versión incluye varios cambios incompatibles con versiones anteriores.

2. Incompatibilidades con la versión OL5

2.1. Uso de map.forEachLayerAtPixel

Debido a consideraciones de rendimiento, las capas en un mapa a veces se renderizarán en una lienzo único en lugar de elementos separados.

Esto significa que map.forEachLayerAtPixel mostrará falsos positivos.

La solución más fácil para evitar eso es asignar diferentes propiedades className a cada capa de esta manera:

```
new Layer({
  // ...
  className: 'my-layer'
})
```

Tenga en cuenta que esto puede incurrir en una pérdida de rendimiento significativa cuando se trata de muchas capas y / o apuntar a dispositivos móviles.

2.2. Eliminación de reglas de impresión CSS

Las reglas de impresión de medios CSS se eliminaron del archivo ol.css. Para obtener el comportamiento anterior, use el siguiente CSS:

```
@media print {
  .ol-control {
    display: none;
  }
}
```

2.3. Los métodos setCenter, setZoom, setResolution y setRotation en ol / View ya no ignoran las restricciones

Anteriormente, estos métodos permitían establecer valores que eran inconsistentes con las restricciones de vista dadas.

Este ya no es el caso y todos los cambios en el estado de la vista ahora siguen la misma lógica: Se proporcionan valores objetivo y se aplican restricciones sobre ellos para determinar los valores reales que se utilizarán.

2.4. Eliminación de la opción restrictinResolution en View.fit, PinchZoom, MouseWheelZoom y ol / interaction.js

La opción constrainResolution ahora solo es compatible con la clase View. También se agregó un método View.setConstrainResolution.

En general, la responsabilidad de aplicar restricciones de centro / rotación / resoluciones se trasladó de las interacciones y controles a la clase View.

2.5. La opción de view extent ahora se aplica a todo el viewport del canvas

Anteriormente, estas opciones solo limitaban el centro de visualización. Este comportamiento aún puede obtenerse especificando constraintCenterOnly en las opciones de vista.

Como efecto secundario, el método rotate de vista eliminado y se ha reemplazado con adjustRotation que toma un delta como entrada.

2.6. Eliminación de métodos obsoletos.

Se ha eliminado la función inherits que se usó para heredar los métodos prototipo de un constructor a otro. Las clases estándar ECMAScript se deben utilizar en su lugar.

Se han eliminado las funciones obsoletas getSnapToPixel y setSnapToPixel de la clase ImageStyle.

2.7. Prerender y postrender sustituyen a precompose y postcompose en los eventos de dibujo de capas

Si anteriormente se estaba registrando precompose y postcompose, ahora deben registrarse los eventos de prender y postrender en capas. En lugar del evento de render, ahora debe escuchar el postrender. Las capas ya no están compuestas por un solo elemento Canvas. En cambio, se agregan a la vista del mapa como elementos individuales.

2.8. La nueva función getVectorContext proporciona acceso a la API de representación vectorial inmediata

Anteriormente, los eventos de render incluían una propiedad vectorContext que le permitía representar características o geometrías directamente en el mapa. Esto todavía es posible, pero ahora debe crear explícitamente un contexto vectorial con la función getVectorContext. Este cambio hace que la API de representación inmediata sea una dependencia explícita si su aplicación la usa. Si no usa esta API, su paquete de aplicaciones no incluirá los módulos de representación vectorial (como lo hizo antes).

Aquí hay un ejemplo abreviado de cómo usar la función getVectorContext:

```
import {getVectorContext} from 'ol/render';


// construct your map and layers as usual


layer.on('postrender', function(event) {
  const vectorContext = getVectorContext(event);
  // use any of the drawing methods on the vector context
});
```

2.9. Las capas solo se pueden agregar a un solo mapa

Anteriormente, era posible renderizar una sola capa en dos mapas. Ahora, cada capa solo puede pertenecer a un solo mapa (de la misma manera que un solo elemento DOM solo puede tener un padre).

2.10. El OverviewMap requiere una lista de capas.

Debido a la restricción anterior (las capas solo se pueden agregar a un solo mapa), el mapa general debe construirse con una lista de capas.

2.11. Cambio de nombre de getGetFeatureInfoUrl a getFeatureInfoUrl

El getGetFeatureInfoUrl de ol/source/ImageWMS y ol/source/TileWMS ahora se llama getFeatureInfoUrl.

2.12. Cambio del comportamiento de los métodos clear() y refresh() de ol/source/Vector

El método ol/source/Vector#clear() ya no activa una recarga de los datos del servidor. Si anteriormente usabas clear() para volver a buscar desde el servidor, ahora tienes que usar refresh().

El método ol/source/Vector#refresh () ahora elimina todas las features del source y desencadena una recarga de los datos del servidor. Si anteriormente estaba utilizando el método refresh() para volver a representar una capa vectorial, debería llamar a ol/layer/Vector#changed().

2.13. Cambio del comportamiento de los métodos clear() y refresh() de ol/source/Vector

El método ol/source/Vector#clear() ya no activa una recarga de los datos del servidor. Si anteriormente usabas clear() para volver a buscar desde el servidor, ahora tienes que usar refresh().

El método ol/source/Vector#refresh () ahora elimina todas las features del source y desencadena una recarga de los datos del servidor. Si anteriormente estaba utilizando el método refresh() para volver a representar una capa vectorial, debería llamar a ol/layer/Vector#changed().

2.14. Cambio del comportamiento del evento precompose y postcompose de las capas

El comportamiento del evento precompose y postcompose ha cambiado, para desarrollar una lógica similar, se deben usar los eventos prerender y postrender en la implementación de las capas.

2.15. Cambios al establecer la opacidad de las capas

Con la nueva versión de OpenLayers la opacidad se debe establecer en formato número, ya no se puede establecer la opacidad como cadena.

2.16. Los niveles de zoom no son enteros por defecto

Cuando se crea un mapa con OpenLayers 6 los niveles de zoom no son enteros por defecto, ahora para simular el comportamiento de la versión anterior, se debe establecer el parámetro “constraintResolution” a true.

2.17. El tamaño del mapa desde el viewPort no es preciso

Para obtener el tamaño del mapa en píxeles deberemos usar el método getSize() de la implementación del mapa en lugar de obtenerla a través del viewport.

2.18. Problemas a la hora de parsear features desde URL en WFS

Al incrementar la versión de openlayers hemos observado comportamientos extraños a la hora de parsear las features desde url de WFS. En la anterior versión al parsear las features le debíamos indicar implícitamente la versión de GML que queríamos. En la nueva versión no es necesario:

Anterior versión:

```
const formatter = new ol.format.WFS({ gmlFormat: ol.format.GML2() });
```

Nueva versión:

```
const formatter = new ol.format.WFS();
```

2.19. OverviewMap requiere que se le establezca la vista directamente

Con la nueva versión de OpenLayers, el control OverviewMap requiere el parámetro view para solucionar problemas cuando el modo de zoom sea fijado.

2.20. Cambios en la interacción Select de OpenLayers

Con la nueva versión de OpenLayers, ol.interaction.Select se comporta de manera diferente. Si hay una ol.Feature seleccionada y se deselecciona elimina todos los cambios de estilo que ha tenido durante el tiempo que ha estado seleccionado.

2.21. Las capas de OpenLayers no tienen el atributo ‘type’

Con la nueva actalización la implementación de capas carecen del atributo ‘type’ lo que no permite identificar de manera directa si una capa es un ‘VECTOR’, un ‘TILE’, una ‘IMAGE’, ….

No funciona correctamente el método setResolutions de la fachada

Con la nueva actalización, para establecer unas resoluciones del mapa personalizadas, se debe utilizar el método de la vista de la implementación del mapa. Ejemplo:

mapJs.getMapImpl().getView().setResolutions([...])

2.22. Undefined en atributos personalizados

Con la nueva actualización, al consultar atributos que no existen en una capa en lugar de devolver null como la versión anterior, devuelve undefined.

2.23. Cambios a la hora de crear Features de Mapea a través de Features de OpenLayers

Al crear esta Feature de manera manual, copiando los atributos de la Feature de OpenLayers puede provocar comportamientos no deseados(pérdida del paralelismo, elementos duplicados, …), para evitar este comportamientos deberemos usar siempre M.impl.Feature.olFeature2Facade y si necesitamos incluir algo extra, se lo incluimos una vez se ha obtenido la Feature de la fachada.

2.24. Se ha eliminado el render mode ‘vector’ de las vector tile layers

Al crear una capa VectorTile si se usaba el renderMode: ‘vector’ se debe eliminar ya que con esta versión no es compatible, ahora se deben usar los modos ‘hybrid’ o ‘image’.

3. Adaptaciones de plugins y visualizadores

Como consecuecia de los cambios de versión han sido realizado los siguientes cambios en plugins

- Tarea #184926: [Attributions] [APIIGN4] Fallos de escala y zoom
- Tarea #184927: [Buffer][APIIGN4] Problemas al cargar el plugin Buffer
- Tarea #184929: [ComparePanel][APIIGN4] Problemas con el plugin ComparePanel
- Tarea #184930: [APIIGN4] Error al intentar modificar opacidad
- Tarea #184932: [IberpixCompare][APIIGN4] Al dividir el mapa se muestran
- Tarea #184935: [GetFeatureInfo][APIIGN4] No funciona GetFeatureInfo
- Tarea #184938: [PrinterMap][APIIGN4] No aparece el perfil topográfico
- Tarea #184939: [Rescale][APIIGN4] Error aproximación de escala
- Tarea #184940: [Vectors][APIIGN4] Problema con los requisitos de zoom a .
- Tarea #184942: [APIIGN4] El botón más información desaparece
- Tarea #184944: [Mapaalacarta][APIIGN4] Mapaalacarta no carga el Step1 co
- Tarea #184946: [Printer][APIIGN4] Problemas con el printer en Fototeca
- Tarea #188052: [Subida versión de PLUGINS a API-CNIG v4]

4. Historial de Versiones

Versión	Fecha	Autor	Cambio	Aprobado
1.0.0	15/08/2021	Guadaltel		